

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2001-282558

(43)Date of publication of application : 12.10.2001

(51)Int.Cl.

G06F 9/46

(21)Application number : 2000-095537

(71)Applicant : HITACHI LTD

(22)Date of filing : 30.03.2000

(72)Inventor : ONO HIROSHI

SAITO MASAHIKO

KATO SUNAO

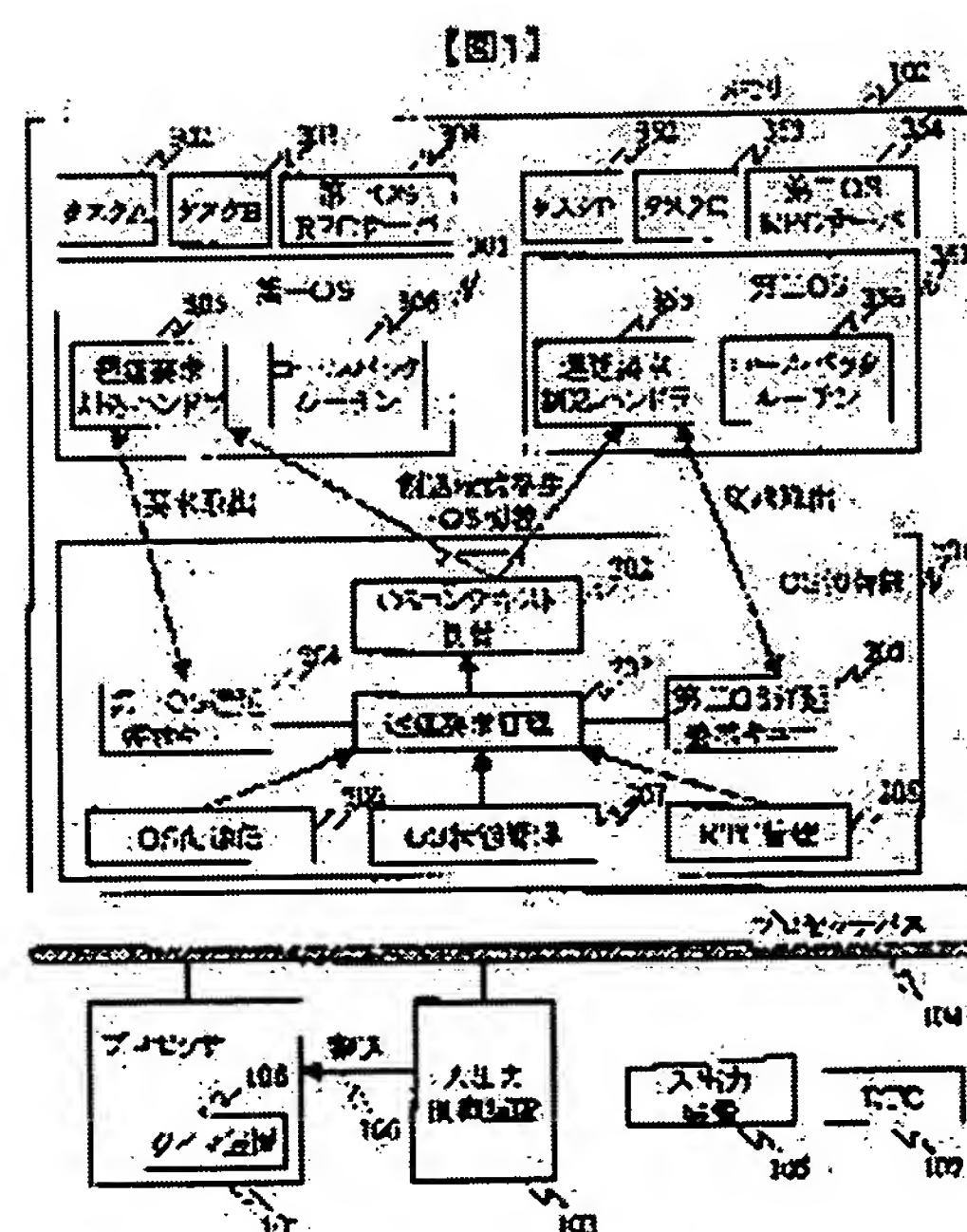
KAMIWAKI TADASHI

(54) MULTI-OPERATING COMPUTER SYSTEM

(57)Abstract:

PROBLEM TO BE SOLVED: To provide a multi-operating computer system realizing the communication of a processing request among a plurality of operating systems with simple modification.

SOLUTION: When the processing request is transmitted by communication between the operating systems 301 and 351, the operating system receiving the processing request interruption-processes the processing request from the other operating system by an interruption handler 305.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than

the examiner's decision of rejection or
application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's
decision of rejection]

[Date of requesting appeal against examiner's
decision of rejection]

[Date of extinction of right]

(19)日本国特許庁 (J P)

(12) 公 開 特 許 公 報 (A)

(11)特許出願公開番号
特開2001-282558
(P2001-282558A)

(43)公開日 平成13年10月12日(2001.10.12)

(51)Int.Cl.⁷

G 0 6 F 9/46

識別記号

3 5 0

F I

G 0 6 F 9/46

テ-71-ト*(参考)

3 5 0 5 B 0 9 8

審査請求 未請求 請求項の数 8 O L (全 26 頁)

(21)出願番号 特願2000-95537(P2000-95537)

(22)出願日 平成12年3月30日(2000.3.30)

(71)出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(72)発明者 大野 洋

茨城県日立市大みか町五丁目2番1号 株
式会社日立製作所大みか事業所内

(72)発明者 齊藤 雅彦

茨城県日立市大みか町七丁目1番1号 株
式会社日立製作所日立研究所内

(74)代理人 100074631

弁理士 高田 幸彦 (外1名)

最終頁に続く

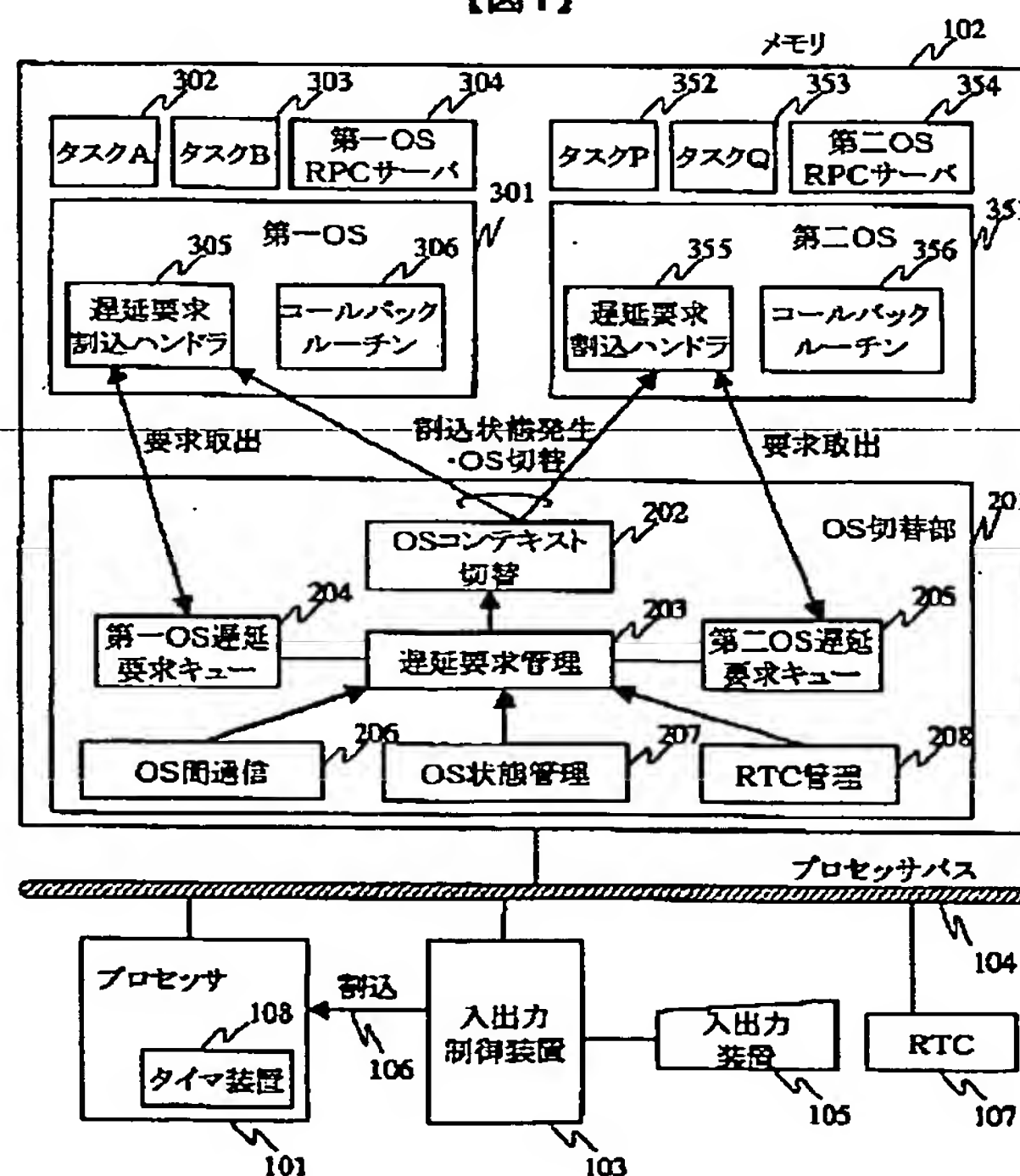
(54)【発明の名称】 マルチオペレーティング計算機システム

(57)【要約】

【課題】本発明の目的は、複数のオペレーティングシステム間の処理要求の通信をオペレーティングシステムの簡単な改造で行えるマルチオペレーティング計算機システムを提供することにある。

【解決手段】本発明は、複数のオペレーティングシステム301、351間で処理要求を通信によって送信する際に、処理要求を受信したオペレーティングシステムは他のオペレーティングシステムからの処理要求を割込ハンドラ305により割込み処理するようにしたことにある。

【図1】



【特許請求の範囲】

【請求項1】複数個のオペレーティングシステムを切替えて一台のプロセッサを動作させ、前記複数個のオペレーティングシステム間で処理要求を行うようにしたマルチオペレーティング計算機システムにおいて、前記処理要求を受けたオペレーティングシステムは他のオペレーティングシステムからの処理要求を割込み処理するようにしたことを特徴とするマルチオペレーティング計算機システム。

【請求項2】複数個のオペレーティングシステムを割込み要求に基づき切替えて一台のプロセッサを動作させ、前記複数個のオペレーティングシステム間で処理要求を行うようにしたマルチオペレーティング計算機システムにおいて、前記処理要求を受けたオペレーティングシステムは他のオペレーティングシステムからの処理要求を割込み処理する割込みハンドラを具備することを特徴とするマルチオペレーティング計算機システム。

【請求項3】複数個のオペレーティングシステムを割込み要求に基づき切替えて一台のプロセッサを動作させ、前記複数個のオペレーティングシステム間で処理要求を行うようにしたマルチオペレーティング計算機システムにおいて、前記オペレーティングシステムが要求する処理要求を一時的に遅延要求キューに格納し、前記処理要求を受けたオペレーティングシステムは他のオペレーティングシステムからの処理要求を前記遅延要求キューから取出す遅延要求割込みハンドラを具備することを特徴とするマルチオペレーティング計算機システム。

【請求項4】割込み要求を発生して複数個のオペレーティングシステムから一つを選択して動作する一台のプロセッサと、前記プロセッサの割込み要求に基づき前記複数個のオペレーティングシステムを切替える切替手段と、前記複数個のオペレーティングシステム間における処理要求を通信する通信手段とを具備し、前記オペレーティングシステムは、他のオペレーティングシステムからの処理要求を割込み処理するようにしたことを特徴とするマルチオペレーティング計算機システム。

【請求項5】割込み要求を発生してオペレーティングシステムを選択して動作する一台のプロセッサと、前記プロセッサの割込み要求に基づき複数個のオペレーティングシステムを選択して切替える切替手段と、前記複数個のオペレーティングシステム間における処理要求を通信する通信手段とを具備し、前記オペレーティングシステムは、他のオペレーティングシステムからの処理要求を割込み処理する割込みハンドラを具備することを特徴とするマルチオペレーティング計算機システム。

【請求項6】割込み要求を発生してオペレーティングシステムを選択して動作する一台のプロセッサと、前記プロセッサの割込み要求に基づき複数個のオペレーティングシステムを選択して切替える切替手段と、前記複数個のオペレーティングシステム間における処理要求を通信

する通信手段と、前記オペレーティングシステムが送信した処理要求を一時的に格納する遅延要求キューとを具備し、前記処理要求を受信したオペレーティングシステムは、他のオペレーティングシステムからの処理要求を前記遅延要求キューから取出す遅延要求割込みハンドラを具備することを特徴とするマルチオペレーティング計算機システム。

【請求項7】複数個のオペレーティングシステムを切替えて一台のプロセッサを動作させ、前記複数個のオペレーティングシステム間で処理要求を通信によって送信するようにしたマルチオペレーティング計算機システムにおける前記処理要求を受信したオペレーティングシステムが他のオペレーティングシステムからの処理要求を割込み処理するようにしたプログラムを格納していることを特徴とする記憶媒体。

【請求項8】複数個のオペレーティングシステムを割込み要求に基づき切替えて一台のプロセッサを動作させ、前記複数個のオペレーティングシステム間で処理要求を通信によって送信するようにしたマルチオペレーティング計算機システムにおける前記オペレーティングシステムが送信した処理要求を一時的に遅延要求キューに格納し、前記処理要求を受信したオペレーティングシステムが他のオペレーティングシステムからの処理要求を前記遅延要求キューから取出させるように処理するプログラムを格納していることを特徴とする記憶媒体。

【発明の詳細な説明】**【0001】**

【発明の属する技術分野】本発明は複数個のオペレーティングシステムを切替えて一台のプロセッサを動作させるマルチオペレーティング計算機システムに関する。

【0002】

【従来の技術】一つの計算機システムで複数のオペレーティングシステム（以下、「OS」と呼ぶ）を動作させる技術として、従来から大型計算機において使用されている「仮想計算機（Virtual Machine。以下、「VM」と呼ぶ）」が知られている。VMの技術では、計算機システム全体を管理するVMコントロールプログラムが存在し、当該計算機システムのハードウェアを独占して管理する。

【0003】計算機システム上で動作する各OSに対して、あたかもハードウェアを独占して管理しているのと同様の状態を提供するために、VMコントロールプログラムがハードウェアのエミュレーションを実行する。また、ハードウェア通信路のエミュレーションも提供され、各OSの間に通信路が存在する。

【0004】複数のOSにまたがって通信を行う場合または複数のOSにまたがって同期をとる場合（以下、両者を合わせて「OS間通信」と呼ぶ）には、各OS上で動作するタスクないし各OS上で動作する専用のサブプログラムが、通信路を使って通信を行い実現する。各O

Sでこの通信路に対するアクセスが行われるのは、あらかじめ当該OSが実行されるように割当てられたタイムスロットの間だけである。

【0005】一方、より小規模な計算機システムにおいても、CPU能力の向上により、一つの計算機システムで複数のOSを動作させることが可能となってきた。このような複数OS共存技術として特開平11-149385が挙げられる。この技術では、OS切替プログラムが、各OSごとのレジスタ内容等のコンテキストを退避復旧し、OSを切替える。この際、切替の契機となるハードウェア割込についてはOS切替プログラムが取扱うが、それ以外のハードウェアのアクセスについては、動作中のOSが実際に操作する。さらに、このような複数OS共存技術を使った計算機システムにおいて、OS間通信を実現する方法として、特開平11-85546にある技術が知られている。

【0006】

【発明が解決しようとする課題】従来技術のうち、前者のVM技術の場合、全てのハードウェア処理をエミュレーションしており、またVM間の通信路もハードウェアの通信路を模擬したものとなるため、処理のオーバーヘッドが大きくなる。従って、CPU能力およびメモリ容量等の資源が限られている小規模な計算機システムには適さない。

【0007】一方、後者の従来技術は、OS共存技術の場合、当該技術を適用する計算機システムには次のような課題がある。

【0008】従来技術では、OSを切替えるプログラムが各OSに対して何らかの処理要求を行う場合、当該OS内の処理モジュールを直接呼出すようにしている。当該計算機システム上で動作するOSとして、本来は計算機システム上で単独で動作するOSを元にしてこれを修正しようとする場合には、OS内のタスク切替部分を修正することが必要となる。これは、当該OSのタスク切替に関連する全ての処理を完全に理解することが必要であり、難易度が極めて高いものになる。

【0009】このことを図31に示す従来の計算機システムを用いて具体的に説明する。

【0010】図31において、第一OS301と第二OS351が、OS切替部201により切替えられながら、単一の計算機システム上で動作している。第一OS301上で動作するタスクA302は、第二OS351上で動作するタスクP352からメッセージが送信されるのを待機している。

【0011】タスクPからメッセージが送信されたとき、OS間通信モジュール206は第一OS301に対して、待機しているタスクA302の動作を再開させるよう指示を行う。この際、OSコンテキスト切替モジュール202が動作コンテキストを第一OS301のものに復旧した上、コールバックルーチン306を呼出す。

このときコールバックルーチン306は、タスクA302を再開させる要求をスケジューラ309に行う必要がある。

【0012】しかし、コールバックルーチン306は、第一OS301が単独で動作する場合には存在しないものであり、スケジューラ309は元々関知していない。従って、コールバックルーチン306が呼出される直前に動作していた第一OS301上のタスクのコンテキストを保存する処理や、スケジューラ309が当該タスクを中断したものとして扱うような状態変更処理などを、既存のスケジューラの状態管理と矛盾が無いように新たに追加しなければならない。

【0013】コールバックルーチン306が直接スケジューラ309を操作せず、タスクA302を再開させるためのシステムコール308を呼出す方法も考えられるが、タスク状態を変更するようなシステムコール308の呼出しには様々な制約（例えば、使用するスタックの状態）が伴い、結局OSの内部処理を完全に把握し、その制約を満たす状態を作り出すことが必要になる。

【0014】また、上述の課題を解決するには図32に示すように構成することも考えられる。

【0015】図32においては、コールバックルーチンは要求されたタスクAの再開を直接行わず、要求があったことをOS内の要求テーブル341に記録し、処理を一旦OS切替部201に戻す。第一OS上では周期的に起動されるルーチン342が存在し、このルーチンが要求テーブル341内の要求を取出して、タスクA302の再開を要求するシステムコール308を発行する。この場合、周期起動ルーチン342は、OSに組込むデバイスドライバまたはタスクを想定しており、システムコールを発行するための条件が予め整っていてOSの内部処理を意識する必要はない。

【0016】しかし、当該OSに動作が切替っても直前に動作していたタスクが再開するだけであり、周期起動ルーチン342が実行されるまでは実際の処理が行われず、タスクA302の再開が遅延することになる。このため、複数のOSを小さいオーバーヘッドで切替えて動作させるという本来の目的を達成できなくなる。

【0017】このように、スケジューラはOS内の全体を管理しているので、他の手順因果関係を考慮した上でOS間通信できるように改造する必要があり、大幅なプログラム変更を要することになる。

【0018】本発明の目的は、複数のオペレーティングシステム間の処理要求の通信をオペレーティングシステムの簡単な改造で行えるマルチオペレーティング計算機システムを提供することにある。

【0019】

【課題を解決するための手段】本発明の特徴とするところは、複数のオペレーティングシステム間で処理要求を通信によって送信する際に、処理要求を受信したオペ

レーティングシステムは他のオペレーティングシステムからの処理要求を割込み処理するようにしたことにある。

【0020】本発明は、オペレーティングシステムが処理要求を割込み処理するようにしているので、オペレーティングシステムに通信用のハンドラ機能を一つ追加するだけでよくオペレーティングシステム間の処理要求の通信をオペレーティングシステムの簡単な改造で行える。

【0021】

【発明の実施の形態】以下、本発明の実施例を図面を用いて説明する。

【0022】図1に本発明の一実施例を示す。

【0023】図1において、計算機システムは、プロセッサ101、メモリ102、入出力制御装置103、入出力装置105、実時刻時計(Real Time Clock。以下、「RTC」と呼ぶ)107などから構成される。プロセッサ101、メモリ102、入出力制御装置103、RTC107はプロセッサバス104によって接続される。

【0024】プロセッサ101は複数のオペレーティングシステム(以下、「OS」と呼ぶ)を動作させるためのマイクロプロセッサである。メモリ102は、オペレーティングシステム切替プログラム201(以下、「OS切替部」と呼ぶ)、第一のオペレーティングシステム301(以下、「第一OS」と呼ぶ)、第二のオペレーティングシステム351(以下、「第二OS」と呼ぶ)、第一OS上で動作するタスク302、303、第二OS上で動作するタスク352、353、第二OSからの要求により第一OS上での処理を行う第一OSRPCサーバ304、第一OSからの要求により第二OS上での処理を行う第二OSRPCサーバ354を保持する。これらのプログラム201、301~304、351~354は、プロセッサ101によって読み出されて実行される。

【0025】入出力制御装置103には、プロセッサ101またはメモリ102との間で、データを入力または出力するための入出力装置105が接続される。入出力装置105としては、データを保存するためのディスク装置、画面表示用デバイスであるディスプレイ、ユーザからの入力用デバイスであるキーボード、あるいは他の計算機システムとのデータを交換するためのネットワーク接続装置などがある。どのような入出力装置が接続されるか、あるいは全て接続されないかは、システム構成によって決定される。

【0026】入出力制御装置103は、割込信号線106によってプロセッサ101と接続され、入出力動作完了などを通知することができる。なお、図1では割込信号線106とプロセッサバス104が別の装置であるように記載しているが、実際には、割込信号線106はプ

ロセッサバス104の一部である。プロセッサ101の内部にはタイマ装置108が設けられており、一定周期毎に割込を発生させる。タイマ装置108からの割込は、OS301、351の計時などに使用される。

【0027】なお、プロセッサ101の種類によっては、入出力制御装置103、入出力装置105、RTC107の一部あるいは全部が、プロセッサ101の内部に取り込まれている場合がある。また、タイマ装置108がプロセッサ101の内部には存在せず、プロセッサバス104を介してプロセッサ101の外部に接続される場合もある。

【0028】プロセッサ101は割込信号線106によって通知されるプロセッサ外部からの割込(以下、「外部割込」と呼ぶ)やタイマ装置108などのプロセッサ内部からの割込(以下、「内部割込」と呼ぶ)をマスクできる機能を有している。割込マスクとは、プログラムが割込マスクを解除するまでの間、特定の割込が入ることを遅延させる機能である。

【0029】一般に、割込マスク機能には、次の三種類が存在する。

(1) 全割込マスク：全ての割込をマスクする。

(2) 個別割込マスク：個々の割込をそれぞれマスクする。

(3) 割込レベルマスク：各割込にレベルを設定し、指定レベル以下の割込をマスクする。

【0030】プロセッサ101の種類によって、上記

(1) および(2)の組合せ、または(1)および

(3)の組合せのいずれかを装備することが多い。後者の組合せを装備するプロセッサを採用する場合、対応する割込要因の重要性にしたがって割込レベルを割当てる。例えば、リアルタイム性が高いネットワーク入出力装置からの割込を、他のディスク装置などの割込よりも高いレベルに設定する計算機システムの構成がある。

【0031】本実施例では、計算機システム内に二つのOS301、351が存在する。OS301、351は、各々に割当てられたメモリ・プロセッサ資源を用い、それぞれプログラム302~304、プログラム352~354を実行させる。RPCサーバ304ないし354は、実際には当該OS上の二つのタスクから構成される。このように実施例としてはOS数が2、タスク数が8(各OSに対して4ずつ)の例を示しているが、これらの数値よりも多い、または、少ないOSないしタスクを実装することも可能である。OS数は動的にタスク生成・削除を行うことにより可能である。

【0032】なお、動作している複数のOSを全て同等のものとして扱っている。従って、以下で単一のOSについてのみしか説明していない部分について、他のOSでも同様の構成を持っている、ないし同様の動作が行われるものとする。

【0033】各OS301、351には、OS切替部2

01の内部から直接呼出されて要求された処理を完結しOS切替部201にリターンする、コールバックルーチン306ないし356が存在する。OS切替部201は各OS301、351に対する処理要求のうち、当該処理要求を即時に実行することが必要なものに関して、コールバックルーチンの呼出を行う。処理要求の例として、当該OSを強制停止させる直前に当該OSが管理する入出力装置からの割込発生を抑止する処理がある。

【0034】OS切替部201には、OSコンテキスト切替モジュール202が設けられており、OS切替部201内の各モジュールからの指示によって、プロセッサ101上で動作するOSを切替える。また、OS切替部201には、OS切替部201から各OS301、351に対する処理要求を管理する遅延要求管理モジュール203があり、第一OS301に対する処理要求を蓄積するために第一OS遅延要求キュー204、第二OS351に対する処理要求を蓄積するために第二OS遅延要求キュー205を有している。処理要求の例として、OS切替部が提供するOS間メッセージ通信機能を使用しメッセージが送られてくるのを受信待機しているタスクがある場合、メッセージが送信された時点で当該タスクの動作を再開させる処理がある。

【0035】遅延要求管理モジュール203は、第一OS遅延要求キュー204ないし第二OS遅延要求キュー205に処理要求が蓄積されていて、なおかつ当該OSが割込を処理するのに適切な条件が整った場合、OSコンテキスト切替モジュール202を通じて当該OSに割込が発生したのと同じ状態とし、当該OSに遅延要求の存在を通知する割込を発生させる。以下、遅延要求管理モジュール203経由で行われる処理要求を「遅延要求」、遅延要求管理モジュールの存在を通知する割込を「遅延要求割込」と呼ぶ。各OS301、351上には、遅延要求割込を受けて、遅延要求キュー204ないし205から要求を取出し対応する処理を行う、遅延要求割込ハンドラ305ないし355が設けられている。

【0036】OS間通信モジュール206は、異なるOSのタスク間でのメッセージを交換する機能（以下、「OS間メッセージ通信」と呼ぶ）や、異なるOS間で同期を取るための機能（以下、「OS間セマフォ」と呼ぶ）を提供する。OS間メッセージ通信機能では、メッセージがまだ存在せず送信されるのを待つためにタスクの動作が中断する場合があります、またOS間セマフォ機能では、獲得しようとしたセマフォが先に獲得されていて当該セマフォが解放されるのを待つためにタスクの動作が中断する場合があります。いずれの場合も、待っている条件が整い次第、当該タスクの動作を再開する必要がある。OS間通信モジュール206は前記の条件が整った場合、タスク再開の遅延要求を追加するよう遅延要求管理モジュール203に対して依頼する。

【0037】OS状態管理モジュール207は、各OS

301、351の起動や停止等の状態を管理し、また状態の移行要求を受付けて各OS301、351に指示する機能を有している。例えば、あるタスクからあるOSに対するシャットダウン要求を受付けた時は、当該OSに対するシャットダウン処理要求を遅延要求として追加するよう、遅延要求管理モジュール203に依頼する。

【0038】RTC管理モジュール208は、RTC107からの時刻取得処理およびRTC107への時刻設定処理を行う。各OS301、351は、RTC107に対して直接時刻を読み書きする代りに、RTC管理モジュール208にこれらの処理要求を行う。例えば、一方のOS301からRTC107への時刻設定処理依頼があった場合、他のOS351に対してRTC107に時刻を合わせて修正するよう遅延要求を追加する。

【0039】各OS301、351のRPCサーバ304、354は、それぞれ、他のOSからOSメッセージ通信機能を使用して送られてくる処理要求を受付け、当該OS上で処理を実行し、さらにOSメッセージ通信機能を使用して処理結果を送り返す。

【0040】図2に、プロセッサ101の内部構成の一例を示す。

【0041】図2において、キャッシュメモリ122はメモリ102上のデータまたは命令を一時的に格納するバッファ記憶装置である。CPU121は演算手段であり、メモリ101もしくはキャッシュメモリ122上に存在する命令を順次読み出して実行する。命令実行に際して、演算結果を一時的に保持するための汎用レジスタ123、命令アドレスを指定するプログラムカウンタ（以下、「PC」と呼ぶ）124、実行状態を保持するステータスレジスタ（以下、「SR」と呼ぶ）125を用いる。

【0042】タイマ装置108として、三つのタイマ装置1081～1083を備えている。タイマ装置108の個数は、後述の割込処理の説明のために三つとしているが、本発明の実施にあたっては最低一つ存在していればよい。

【0043】割込信号線106とタイマ装置108からの割込信号は割込コントローラ126に接続される。割込コントローラ126は、外部割込または内部割込が発生した場合、SR125の内部にある割込マスクに関する情報を参照し、当該割込を受け付けるか否かを決定する。

【0044】割込を受付ける場合、割込コントローラ126は、割込の要因を割込要因レジスタ130に記録し、PC124の現在値を退避プログラムカウンタ（以下、「SPC」と呼ぶ）126に保存してから割込アドレスレジスタ129の値に書換える。さらに、SR125の現在値を退避ステータスレジスタ（以下、「SSR」と呼ぶ）128に保存してから書換える。SR125の書換え内容については後述する。この結果、割込ア

ドレスレジスタ130に登録されたアドレスに存在する、割込処理プログラムが起動する。割込処理プログラムは割込要因レジスタ130に記録された割込要因を参照して、これに対応する処理を実行する。

【0045】なお、使用するプロセッサ101の種類により、割込処理プログラムのアドレスをメモリ上に格納し、当該格納位置のアドレスをプロセッサ内のレジスタに設定することもできる。

【0046】また、使用するプロセッサ101の種類により、割込要因レジスタ130に割込要因を示す値を設定する代りに、割込要因ごとに異なる割込処理プログラムのアドレスを使用することもできる。この場合、各割込要因ごとの割込処理プログラムで要因を示す値を特定の汎用レジスタに記録してから共通の割込処理プログラムにジャンプし、当該共通割込処理プログラムで割込要因レジスタ130に記録された要因を参照する代りに前記の汎用レジスタを参照するようにすれば、図2で示したプロセッサと全く同様の割込処理を行うことができる。

【0047】CPU121、キャッシュメモリ122、各レジスタ123～125、127～130は、互いに、データ転送を行うデータバス131、アドレス指定を行うアドレスバス132によって接続されている。

【0048】図3は、プロセッサ101が全割込マスク機能と割込レベルマスク機能とを装備している場合の、SR125の構成例である。SR125は、割込ブロックビット141と割込マスクレベルフィールド142を有する。割込ブロックビット141がONの場合、プロセッサ101に対する全ての割込がマスクされる。割込マスクレベルフィールド142は現在の割込マスクレベル値を示し、これ以下の割込レベルの割込は受け付けられない。

【0049】図3の例では、割込マスクレベルフィールド142は4ビット長であり、合計16種類のマスクレベルを指定可能である。通常、割込レベル0は「割込が発生していない」ことを意味し、割込マスクレベルを0に設定すると「割込マスクを行わない」という指定になるため、実質的には15種類となる。

【0050】プロセッサ101の割込マスクレベルフィールド142のビット数を変更することにより、使用できる割込レベルの種類を増減させることができる。特権モードビット149がONの場合、のプログラムは特権モードで動作しており、プロセッサ101に対する全ての操作が可能である。一方、特権モードビット149がOFFの場合、動作中のプログラムは非特権モードで動作し、一部の機能、例えば、制御用レジスタやメモリ管理レジスタへのアクセスなどが禁止される。

【0051】一般に、OSは特権モードで動作する。一方、OS上で動作するタスクは、OSの種類により、特権モードで動作するものや非特権モードで動作するもの

がある。

【0052】図4は、プロセッサ101が全割込マスク機能と個別割込マスク機能とを装備している場合の、SR125の構成例である。SR125は実際には二つのレジスタ（実行状態レジスタ143と割込マスクレジスタ144）から構成され、SSR127もこれら二つのレジスタを格納する領域を備える。図3と同様、実行状態レジスタ143内に割込ブロックビット141が設けられている。割込マスクレジスタ144内の割込マスクビット145～148はそれぞれ別々の割込に対応しており、割込マスクビットのいずれかをONとした場合、当該ビットに対応する割込が受け付けられなくなる。

【0053】図3に示すSRの例は、図4のSRの特殊例とみなせる。たとえば、割込マスクビット145のみONとなっている状態をレベル1とし、割込マスクビット145、146の二つがONとなっている状態をレベル2、割込マスクビット145～147の三つがONとなっている状態をレベル3、…、のように対応させることができる。このため、以降、ステータスレジスタ125は図4に示す構成を有するとして説明する。

【0054】割込を受理した場合、割込コントローラ126によりSR125が書換えられる。一般的なプロセッサでは、割込ブロックビット141および特権モードビット149がONに書換えられる。

【0055】OS、デバイスドライバ、および特権モードで動作するタスクは、割込ブロックビット141や割込マスクレジスタ144の内容を書き換えることが可能で、割込マスクレベルを設定できる。これにより、割込処理実行中に同一割込やより優先度の低い割込が発生することを禁止したり、あるいは排他制御を実現することができる。

【0056】以下、その動作について詳細に説明する。

【0057】本実施例では、動作している複数のOSを全て同等のものとして扱っている。従って、以下の説明で一つのOS上での動作、または一つのOSから他の一つのOSへの動作しか説明しない部分があるが、これらの処理を他のOSに対しても同様に適用することが可能である。

【0058】図5に詳細なソフトウェアの機能ブロック図を示す。ただし、一部のコンポーネントについては図5に図示しておらず、図6～図8に示している。図5～図8に記載されている全てのコンポーネントはメモリ102上に格納されており、プロセッサ101にて実行される。以下の説明では、計算機システム上でのいくつかの動作パターンに分割して、各コンポーネントの詳細な動作を説明する。なお、図9以降の各図面が各コンポーネントの処理フローならびに処理で使用するデータ構造を示している。

【0059】第一に、各OS上のタスクが当該OSのスケジューリング機能により切替えながら動作している場

合の動作について説明する。

【0060】第一OS301内には、タスクに対してサービスを提供するシステムコールプログラム308、およびタスク切替を行うスケジューラ309を有する。また、第一OS301上で動作するタスクがOS切替部201の処理を利用するための関数を格納したOS切替部利用ライブラリ310を有する。

【0061】図5では、OS切替部ライブラリ310をタスクA302が利用している構成を示しているが、他のタスクからも利用可能である。タスクA302は、OS切替部ライブラリ310が提供する関数を呼出すことで処理を依頼する。呼出されたOS切替部ライブラリ310内の関数は、タスクA302の一部として動作し、OS切替部201の処理関数を呼出したり、必要なシステムコール308を呼出したりして、依頼された処理を実現する。

【0062】なお、タスクが専用の論理メモリ空間で動作している、あるいはタスクが非特権モードで動作しているなどの理由により、タスクの一部として動作するOS切替部ライブラリ310が直接OS切替部201の関数の呼出することができない場合、OS切替部利用ライブラリの機能の一部を、デバイスドライバなどの形でOS内に置けばよい。

【0063】スケジューラ309は、第一OS301上に存在する各々のタスクに対応してタスク管理ブロック321をメモリ上に確保し、各タスクを管理する。

【0064】タスク管理ブロックの内部構造例を図9に、タスク管理ブロックの管理構造例を図10に示す。タスク管理ブロック321の内部には、複数のタスク管理ブロックをキューに接続して管理するためのキュー接続用ポインタ3211、当該タスクの状態を表す状態フラグ3212、当該タスクが中断した時点のPC、SR、汎用レジスタ値等を保存するためのレジスタ退避領域3213を含む。また、その他に中断理由や優先順位等の情報を含む場合もある。

【0065】スケジューラ309は、実行可能なタスクに対応するタスク管理ブロックを、優先順位毎の実行可能タスクキュー（322、323など）に接続して管理する。従って、実行可能タスクキューは、優先順位の数だけ存在することになる。ただし、優先度の高いタスクのタスク管理ブロックをより前方に接続するという規則を追加することにより、単一の実行可能タスクキューにより管理することも可能である。

【0066】入出力装置105に対する処理完了待ち、あるいはミューテックス・セマフォ・イベントなど第一OS301内の同期処理メカニズムによる再開待ちといった理由で処理を中断しているタスクに対応するタスク管理ブロックを、実行中断タスクキュー324に接続して管理する。なお、実行中断理由ごとに異なるキューで管理することもある。これらの各キューにタスク管理ブ

ロックを接続するためにキュー接続用ポインタ3211が使用される。

【0067】スケジューラ309は、システムコール308の処理や後述する割込ハンドラ307の処理が終了した時点で、現在実行中のタスクが実行可能でなくなった場合または現在実行中のタスクより高い優先順位のタスクが実行可能となった場合に、実行タスクを切替えるために呼出される。スケジューラ309を起動する可能性のあるシステムコールには、（ア）タスクの生成・終了・停止・再開（たとえば、自分より優先順位の高いタスクを生成した場合）、（イ）排他制御の実行・終了（たとえば、排他制御待ち状態に移行した場合）（ウ）タスクの優先順位変更（たとえば、自己の優先順位を低下させた場合）といったものがある。

【0068】スケジューラ309の処理フローを図11に示す。スケジューラ309は、実行中だったタスクのレジスタを当該タスクのタスク管理ブロック内にあるレジスタ退避領域3213に保存し（処理401）、実行可能タスクキューを優先順位が高い方から順に検索する（処理402）。

【0069】タスク管理ブロックが一つも無かった場合は、処理403で判定され、OS間優先順位管理モジュール211にアイドル状態であることを通知する（処理405）。そして実行可能タスクが存在しないのであるから、アイドルループへ移行し（処理408）、実行すべきタスクが現れるまで何も行わない。

【0070】一方、タスク管理ブロックが存在した場合には、当該タスクの優先順位をOS間優先順位管理モジュール211に通知する（処理404）。ただし、通知する優先順位は全てのOSに渡って統一的に決められた優先順位を使用するものとし、第一OS301内で管理されている優先順位から予め決められた対応関係により変換して通知する。

【0071】OS間優先順位管理モジュール211では、ここで（ア）通知した優先順位と、（イ）他のOSで当該OSが中断する前に動作していたタスクの優先順位とを比較する。（以下、（ア）（イ）をまとめて、単に「当該OSの優先順位」と呼ぶ。）通知した優先順位が、第二OS351の優先順位よりも高いか同じだった場合、処理404はすぐにリターンする。

【0072】一方、通知した優先順位が第二OS351の優先順位よりも低かった場合、この時点で第一OS301の実行が中断され第二OS351に切替わり、その後第二OS351で動作しているタスクの優先順位が低下して第一OS301の優先順位の方が高くなった時点で処理404が終了し戻ってくる。ここで選択したタスク管理ブロックをキューから取出し（処理406）、レジスタ退避領域3213の内容をプロセッサ101の各レジスタに復旧し（処理407）、選択したタスクの処理が再開される。なお、当該タスクが新規作成された場

合はレジスタ退避領域3213には初期レジスタ値が登録されており、同じ処理でタスクの新規起動が行われる。

【0073】OS間優先順位管理モジュール211の処理フローを図12に示す。OS間優先順位管理モジュール211は、各OSが最後に通知した優先順位を記録する変数を持つ。この変数が保持する値は、前述の当該OSの優先順位に相当するため、以下、この変数を「OS優先順位変数」と呼ぶ。OSのスケジューラ309ないし359から優先順位の通知を受けると、当該OSに対応するOS優先順位変数の内容を通知された値に更新し（処理411）、全てのOS優先順位変数を比較してもっとも高い優先順位を持つOSを次に動作するOSとして選択する（処理412）。

【0074】なお、複数のOS優先順位変数が同一の最高値を持つ状態になった場合、当該OS優先順位変数に対応するOSのうち任意のものを選択すればよい。また、これらの複数のOSの中に最後に優先順位を通知したOSが含まれる場合には、切替オーバヘッドを削減するために当該OSを選択するものとする。

【0075】次に動作するOSを選択した後、遅延要求管理モジュール203を呼出す（処理413）。遅延要求が存在する場合の処理は後述するので、ここでは遅延要求が存在しなかったものとして説明を続ける。処理412で選択したOSが最後に動作していたOSだった場合、すなわち最後に優先順位を通知したOSの優先順位が最も高かった場合には、処理413がリターンしてくるので、呼出元であるスケジューラ309ないし359にそのままリターンする（処理414）。この結果、当該OSのスケジューラが再開して選択したタスクが実行される。

【0076】一方、処理412で選択したOSが最後に動作していたOSと異なる場合、遅延要求管理モジュール203はOSコンテキスト切替モジュール202を呼出して動作OSを切替える。処理413はリターンしない。なお、いずれかのOSのスケジューラ309、359がOS優先順位管理211の呼出を行わない構成とした場合、当該OSに対応するOS優先順位変数を固定値とすることで、当該OS上の全てのタスクが前記固定値の優先順位で動作しているものとみなしてOS間の優先順位を管理することが可能である。

【0077】OSコンテキスト切替モジュール202の処理フローを図13に示す。OSコンテキスト切替モジュール202は、OS切替部201内の他のコンポーネントから呼出され、動作するOSを切替え、必要に応じて割込と同じ状態を作り出す。OSコンテキスト切替モジュール202はメモリ102上に各OSごとのコンテキスト退避領域を確保し、管理している。ここでは、OS間優先順位管理モジュール211からのOS切替要求を処理する部分についてのみ説明する。

【0078】図13の処理421ならびに処理422はいずれもNOと判定され、プロセッサ101の各レジスタの現在値を直前に動作していたOS（以下、「切替元OS」と呼ぶ）のコンテキスト退避領域に保存し（処理425）、これから動作させようとしているOS（以下、「切替先OS」と呼ぶ）のコンテキスト退避領域にある値をプロセッサ101の各レジスタに復旧する（処理426）。その後PCも保存されていた値に復旧し（処理428）、切替先OSの中断点から動作が再開する。

【0079】処理425をより詳細に説明する。OS切替部201に入ってからOS切替部201内の各モジュールで変更したレジスタについては、現在値ではなくOS切替部201に入った時点の値を保存する。これにより、当該OSの動作が再開された時点において、OS切替部201に切替る前の状態を完全に復旧できる。

【0080】また、OS切替部201内のモジュールを関数呼出の形で明示的に呼出した結果で処理425を実行する場合には、PCの値として当該呼出関数からのリターンアドレスを保存し、リターン値を示すレジスタの値として当該呼出関数のリターン値を保存し、その他のレジスタについては前述のとおりOS切替部201に入った時点の値を保存することがある。これにより当該OSの動作が再開された時点において当該呼出関数がリターンした状態から実行されることになる。

【0081】以上で説明した動作により、各OS上のタスクが、全てのOSに渡って統一的に決められた優先順位に従って、順次OSを切替えられながら動作していくことになる。

【0082】第二に、割込が入った場合の動作について説明する。ここで説明する割込とは、前述の外部割込や内部割込であり、遅延要求割込については後述する。

【0083】割込を受付けた場合の割込処理プログラムとして、図5に示されている共通割込ハンドラ212が呼出される。すなわち、割込アドレスレジスタ129には共通割込ハンドラ212のアドレスを設定している。割込要因ごとに第一OS301または第二OS351のいずれかで処理するものとして割込を行う。共通割込ハンドラ212がその割込に基づき、OSコンテキスト切替モジュール202に当該OSに対する割込状態の発生を依頼する。

【0084】OSコンテキスト切替モジュール202は、当該OSの割込ハンドラ307ないし357を呼出し、割込処理を実行させる。各割込ハンドラは必要に応じてシステムコールを発行し、その結果スケジューラが実行されてタスクの切替が発生する場合もある。例えば、入出力装置105の処理完了により割込が発生し、当該処理の完了を待機していたタスクの中断が解除されて当該タスクが再開される場合である。また別な例として、タイマ装置108からの割込発生により当該OSの

内部時間が更新され、ラウンドロビン方式のスケジューリングを行うために現在実行中のタスクが中断され他の同一優先順位の実行待ちタスクにタスクスイッチされる場合がある。なお、本計算機システムでは、一部の割込要因をOS切替部201の内部コンポーネントで処理するように定義することもできる。

【0085】割込を受付けた場合には、即座に処理を行うOS上の割込ハンドラ305ないし355、あるいはOS切替部201の内部コンポーネントを呼出す。従って、ある処理の実行中に、当該実行中の処理よりも処理の必要性が低い割込が発生して、その割込ハンドラにより当該実行中の処理が中断されることを防止するためには、割込マスクレベル値を上げることが必要である。

【0086】共通割込ハンドラ212の処理フローを図14に示す。共通割込ハンドラ212は、図15に示す割込割当テーブル231を持っており、割込要因レジスタ131の値と、当該割込要因レジスタ231の値に対応する割込を処理すべきOSとの組合わせを記録している。図15に示した例の場合、割込要因レジスタ131に設定される値は、200（16進数）から始まり、20（16進数）刻みの値であるものとしている。また、以下の説明では、200（16進数）はタイマ装置1「1081」からの割込、220（16進数）はタイマ装置2「1082」からの割込、240（16進数）はタイマ装置3「1083」からの割込を受付けた場合に設定される値とする。ここで、割込割当テーブル231の内容は固定されたものとしているが、各OSや各OS上で動作するタスクからの要求により内容を動的に変更することも可能である。

【0087】共通割込ハンドラ212は、割込発生時点で呼出され、割込禁止で動作する。まず、プロセッサ101の割込要因レジスタ131を参照し、割込要因を取得する（処理441）。次に割込割当テーブル231から当該割込を処理するOSを取得する（処理442）。ここで割当がOS切替部201となっていた場合（処理443）、OS切替部201内の当該割込を処理するモジュールを呼出して処理を実行し（処理445）、終了後に当該処理で使用したレジスタを元に戻してからプロセッサの割込リターン命令（Return from Exception。以下、「RTE」と呼ぶ）により割込発生時点の動作に復帰する（処理446）。

【0088】ここで割込要因とOS切替部201内の処理を担当するモジュール（呼出される関数）との関係は、別途割当表が存在することを想定しているが、割込割当テーブル231の一部として記録することも可能である。一方、割当がOS切替部201ではない場合、当該OSに対して割込発生状態とするよう、OSコンテキスト切替モジュール202へ依頼する（処理444）。

【0089】OSコンテキスト切替モジュール202では、図13に示すように、処理で使用したワークレジス

タの値を、割込が発生した時点の内容に回復する（処理423）。ここで、割込発生時点で動作していたOSと当該OSが異なっていた場合（処理424）、既に説明したように切替元OSのレジスタを退避し（処理425）、切替先OSのレジスタを復旧する（処理426）。続いて、当該OSの割込ハンドラ307ないし357のアドレスをPCに設定することで、当該割込ハンドラの実行を開始する（処理429）。

【0090】図15に示した割込割当テーブル231の内容の場合、タイマ装置1「1081」からの割込が発生した場合は第一OS301に割込を発生させ、タイマ装置2「1082」からの割込が発生した場合は第二OS351に割込を発生させる。また、タイマ装置3「1083」からの割込が発生した場合はOS切替部201内部のOS状態管理モジュール207が呼出されるものとする。OS状態管理モジュール207で行う割込処理の内容は後述する。

【0091】なお、本実施例ではタイマ装置が三つ存在するものとして、各タイマの割込に対応して各OSないしOS切替部201が呼出されるものとしたが、タイマ装置が一つしか無い場合でも、当該タイマ装置からの割込が発生するごとに予め決められた順序で各OSないしOS切替部201のいずれかへの割込として扱うことで同じ効果を得ることができる。

【0092】なお、割込要因レジスタ131に、割込割当テーブル231に存在しない値が設定されていた場合があり得る。また、割込割当テーブル231の割当OSの欄に第四の値として「無効」と設定されている場合があり得る。これらの場合、共通割込ハンドラ212で単純にRTE（処理446）を行い無視するか、またはハードウェアの重大エラーとして扱い、エラー表示、エラー記録、ないし計算機システムの停止などの処理を行うかの、いずれかとする。

【0093】第一OS301の割込ハンドラ307の処理フローを図16に示す。割込ハンドラ307も割込禁止の状態で呼出され、まず現時点のレジスタの内容を割込スタックに退避する（処理451）。割込要因レジスタ131から割込要因を取得し（処理452）、当該割込とレベルが同じかよりレベルの低い割込が発生しないように割込マスクレベルを設定してから割込禁止を解除する（処理453）。処理453以降では、より割込レベルの高い割込を受付けて、さらに割込ハンドラが呼出されて処理すること（これを「多重割込」と呼ぶ）が可能になる。そして、割込要因に対応する各々の処理を実行する（処理454）。

【0094】一般的なOSの場合、処理454の部分についてはデバイスドライバなどの形でユーザの任意の処理を組込めるような仕組みとなっており、処理の組込み方法やシステムコール308ないし358の使用方法等が明確に規定されている。またOSによっては、処理45

4の内容を動作中に動的に変更することも可能である。各処理が終ると再度割込禁止とし（処理455）、多重割込の処理中であった場合（処理456）や、割込処理の結果スケジューリングが発生しなかった場合（処理457）は、処理451で割込スタック上に退避したレジスタを復旧して（処理460）、RTE命令により割込発生時点の処理に復帰する（処理461）。一方、割込処理内でのシステムコール発行などによりスケジューリングが必要となった場合には、処理451で割込スタック上に退避したレジスタをそれまで動作していたタスクのタスク管理ブロック内のレジスタ退避領域3213に保存し、割込スタックを破棄する（処理458）。そしてスケジューラ309へジャンプして割込ハンドラ307の処理を終了する。

【0095】以上で説明した動作により、割込を受付けた場合に、必要に応じてOSを切替えながら、当該割込を割込ハンドラにて処理される。

【0096】第三に、OS間通信、特にOS間メッセージ通信機能の動作について説明する。

【0097】図17にOS間メッセージ通信の概要を示す。OS間メッセージ通信は、OS間通信モジュール206の一機能として提供される。OS間通信モジュール206内に、メッセージキューと呼ばれるバッファ領域が存在する。メッセージキューには、ID番号が付けられ各々を識別している。

【0098】図17には、ID=1のメッセージキュー241と、ID=2のメッセージキュー242の二つが示されている。図17では、タスクP352が、OS切替部ライブラリ360を通じて、メッセージ243を送信している。メッセージの内容はメッセージキュー241に一旦コピーされ、その後、タスクA302がOS切替部ライブラリ310を通じて当該メッセージを取出す（受信）。

【0099】図17のメッセージ244、245のように、メッセージキューは受信要求があるまでの間複数のメッセージを保持することが可能である。この場合、メッセージ受信一回につき、FIFO（First In First Out）方式で一つのメッセージのみを取出して返す。

【0100】ところで、タスクA302のメッセージ受信が、タスクP352のメッセージ送信よりも先に行われた場合、メッセージキュー241には取出すべきメッセージが存在しない、すなわち空の状態である。この場合、タスクA302からの指定により二つの動作が選択可能である。第一はメッセージが存在しないものとしてエラーとする動作である。第二はメッセージが送信されるまでタスクA302の実行を停止し、メッセージが送信された時点で、A302の実行を再開して当該メッセージを取出すという動作である。第二の動作を、以下、「受信待機」と呼ぶ。

【0101】以上のような手順で異なるOS上のタスク間でのメッセージ通信を実現する。なお、ここで説明しているOS間メッセージ通信の機能を使い、同一のOS上で動作するタスク間でのメッセージ通信を行うことも可能である。

【0102】OS切替部ライブラリ310のメッセージ受信処理（処理470）の処理フローを図18に示す。以下の説明では、メッセージキューが空であった場合に受信待機をするよう指定された場合についてのみ説明する。この処理では、第一OS301の同期機能の一つであるイベントを使う。ここで、各イベントはイベントを識別するID（以下、「イベントID」と呼ぶ）を持つものとする。各イベントはシグナル状態と非シグナル状態の二つの状態が存在し、これらの状態を変更するシステムコールが提供されているものとする。また、任意のタスクは任意のイベントについてシグナル状態になるのを待機することができ、待機している間ポーリングなどのCPU処理を一切行うことなく動作が中断しているものとする。

【0103】メッセージ受信処理470では、まずOS間通信モジュール206に対して、メッセージキューのID番号と自分自身のタスクIDを指定して受信要求を行う（処理471）。ここでメッセージキューが空でなければ（処理472）、取出したメッセージの内容とともに呼出元にリターンする（処理473）。OS間通信モジュール206から何らかのエラーが返された場合も同様である。

【0104】一方、メッセージキューが空であった場合、OS間通信モジュール206は処理保留のリターン値を返してくるので、イベントを非シグナル状態で作成し、このイベントIDと自分自身のタスクIDを遅延要求ハンドラ305に登録した上で、イベントがシグナル状態になるまで待機する（処理474）。

【0105】当該イベントはメッセージが送信された場合ないし何らかのエラー状態が発生した場合に、遅延要求ハンドラ305によりシグナル状態に変更される。当該イベントがシグナル状態になった時点で、再度OS間通信モジュール206にメッセージ受信要求を行い、当該メッセージを取出し（処理476）、使用したイベントを削除した上で呼出元にリターンする（処理477）。処理476でエラーが返された場合も同様である。なお、処理476では、当該要求が受信待機していた後の受信要求再発行であることを示す「再要求フラグ」を付ける。

【0106】OS間通信モジュール206では、当該フラグにより受信待機していた処理を、当該IDのメッセージキューに対する他の受信要求と区別し、受信待機していた処理にメッセージを取出させるようにする。処理474のイベントの待機処理では、タイムアウトを設定する。もし、タイムアウトが発生した場合（処理47

5)、OS間通信モジュール206に対して、受信待機している要求が無くなったことを知らせる通知を行い

(処理478)、使用したイベントを削除した上で呼出元には、タイムアウトエラーでリターンする(処理479)。

【0107】なお、以上の説明では、受信待機が発生する毎にイベントを生成、削除するように記したが、あらかじめタスク毎にイベントを作成したり、いくつかのイベントを作成しておき使用されていないものを順次使用するということも可能である。

【0108】OS間通信モジュール206のメッセージ受信処理(処理480)の処理フローを図19に示す。ここでも、メッセージキューが空であった場合、受信待機をするよう指定された場合についてのみ説明する。

【0109】OS間通信モジュール206は、各メッセージキュー毎に、受信待機中フラグと受信遅延要求発行フラグという二つの変数を持っている。このうち受信待機中フラグは、当該メッセージキューが空であったためにメッセージの受信待機を行っているタスクが存在することを示すフラグで、受信待機を行っているタスクの動作OSとタスクIDが格納されている。受信遅延要求発行フラグは、受信待機が行われた後にメッセージ送信され、受信待機しているタスクからの再度の受信要求により当該メッセージが取出されるのを待機している状態であることを示すフラグである。

【0110】まずメッセージ受信要求を受けると、当該メッセージキューの受信待機中フラグを調べ(処理481)、もし待機中ならばエラーリターンとする(処理482)。すなわち、同時に複数のタスクが受信待機をすることはできない。

【0111】次に、受信遅延要求発行フラグが設定されていない場合、当該キューにメッセージが存在するか確認し(処理484)、存在すればFIFO式にメッセージを一つ取出し、当該メッセージを本処理の呼出元が用意したバッファにコピーしてリターンする(処理485)。一方、メッセージが存在しない場合、要求元タスクの動作OSとタスクIDを当該キューの受信待機中フラグに設定し(処理486)、受信待機を受付けたことを示す「処理保留」のリターン値でリターンする(処理487)。

【0112】一方、受信遅延要求発行フラグが設定されている場合、受信待機が行われた後にメッセージ送信されたことを示している。当該受信待機を行っているタスク以外から当該キューに対するメッセージ受信要求があった場合、すなわち受信要求に「再要求フラグ」が付いていなかった場合(処理488)、当該要求はエラーリターンとする(処理490)。一方、当該受信待機を行っているタスクからの受信処理要求(図18の処理476など)があった場合、受信遅延要求発行フラグをOFFした上で(処理489)、前述のメッセージ取出し処

理485を行う。

【0113】OS間通信モジュール206のメッセージ送信処理(処理500)の処理フローを図20に示す。

【0114】まず送信されたメッセージを指定のキューにコピーする(処理501)。次に当該キューの受信待機中フラグを調べ(処理502)、受信待機中でなければそのまま呼出元にリターンする(処理506)。一方、受信待機中フラグが設定されていた場合、当該フラグに記録されているタスクIDを取出した後当該フラグをクリアし(処理503)、当該キューの受信遅延要求割込フラグをONにする(処理504)。そして、受信待機しているタスクの動作を再開させるよう、遅延要求管理モジュール203に、処理503で取出したタスクIDのタスクに対するタスク再開の遅延要求の追加を依頼する(処理505)。

【0115】遅延要求管理モジュール203は、遅延要求を蓄積してすぐにリターンする場合もあれば、OSコンテキスト切替モジュール202を通じて遅延要求割込を発行し遅延要求割込ハンドラへ処理が移行する場合もある。後者の場合、本送信処理を要求したタスクの属するOSが当該タスクの動作を再開する時点で、処理505のリターン時点に復旧するようにPCの退避を行う。これによりいずれの場合でも、呼出元へのリターンが行われる(処理506)。

【0116】遅延要求管理モジュール203の処理フローを図21～図23に示す。

【0117】遅延要求管理モジュール203は、大きく分けて二つの状況で呼出される。第一は新たな遅延要求が発生した場合である。この場合、当該要求先OSに対応した遅延要求キュー204ないし205に要求を一旦格納する(処理512)。第二は、OS切替部201内の各モジュールがOSコンテキスト切替モジュール202にOSの切替を要求する場合で、共通割込ハンドラ212からの割込発生要求を除き、遅延要求管理モジュール203を経由する形での要求とする。この場合処理512はスキップされる(処理511)。

【0118】遅延要求管理モジュール203は、図5に示すように、各OSに対する遅延要求割込許可フラグ209ないし210を持っている。このフラグがOFFの場合は、当該OSに対して一旦遅延要求割込を発行して当該割込に対する遅延要求割込ハンドラの処理が完了していないことを示している。また、遅延要求管理モジュール203は、各OSに対する遅延要求割込レベルを記録する変数を持っていて、予め当該OSに対する遅延要求割込の割込レベルを設定しておく。

【0119】遅延要求管理モジュール203は、まず第一OSの遅延要求割込レベルを現時点でのプロセッサ101の割込マスクレベルと比較し、割込マスクレベルの方が高ければ第一OSに対する以降の処理をスキップする(処理513)。次に第一OSに対する遅延要求割込

許可フラグ209を調べ、これがOFFの場合は第一OSに対する以降の処理をスキップする(処理514)。遅延要求割込ハンドラ305は第一OS遅延要求キュー204内にある遅延要求を全て取出すまで処理を継続するため、当該処理が完了するまでは新たに遅延要求の発生を通知する必要がない。

【0120】従って、無意味な遅延要求割込の発生およびそれに伴うOS切替を抑止するために、処理514の判定を行っている。続いて、第一OS遅延要求キュー204に遅延要求が存在するか調べ(処理515)、要求が存在する場合には第一OS301に対する遅延要求割込許可フラグ209をONにして(処理519)、遅延要求割込ハンドラ305に遅延要求の存在を通知するために遅延要求割込を発生させる(処理520)。

【0121】実際の遅延要求割込の発生はOSコンテキスト切替モジュール202にて行われる。遅延要求割込の発生要求があった場合は、図13に示した処理フローの中の処理422で判定され、共通割込ハンドラからの割込発生要求と同様に扱われる。この際、処理423において、割込要因レジスタ130に対して予め決めておいた遅延要求割込を示す値を設定する。この値は、他の割込要因では設定されない値を使用する。第一OS301と第二OS351で、同一の値とすることも異なる値とすることも可能である。割込ハンドラ307では、割込要因レジスタ130に遅延要求割込を示す値が設定されていることを確認し、割込処理454として遅延要求割込ハンドラ305を起動する。

【0122】さて、遅延要求管理モジュール203の説明に戻る。第一OS301に対して遅延要求割込の発生を行わなかった場合、第二OS351に対しても同様に遅延要求割込を発生させるかどうかの判定を行う(処理516～処理518)。第二OS351に対して遅延要求割込を発生させると判断した場合は、第一OS301と同様に処理519および処理520を行う。

【0123】第一OS301および第二OS351のいずれに対しても遅延要求割込の発生を行わなかった場合、遅延要求管理モジュール203が呼出された理由を処理511と同じ判定で調べ、新たな遅延要求が発生した場合の呼出ならば呼出元にリターンし(処理522)、OSコンテキスト切替モジュール202にOSの切替を要求する場合であればOSコンテキスト切替モジュール202にジャンプする(処理523)。

【0124】ここまでの説明では、遅延要求割込の発生判定を第一OS301、第二OS351の順で固定的に行うものとしたが、他の順序でも構わない。例えば、遅延要求割込の割込レベルが高い方を先に行ったり、遅延要求の追加時は追加された要求のOSを先に行ったりということも考えられる。

【0125】また、遅延要求割込許可フラグの状態、および遅延要求割込の割込レベルと割込マスクレベルの比

較によって割込発生するかどうかを決めているが、他の基準を追加することも可能である。例えば、遅延要求割込を発生させることができるOSの動作優先順位を予め決めておき、当該OSの優先順位の現在値が予め決められた優先順位よりも低いときに限り遅延要求割込を発生させるようにすることができる。

【0126】次に、第一OS301の遅延要求割込ハンドラ305の処理フローを図24に示す。前述のとおり、遅延要求割込ハンドラ305は、割込ハンドラ307の中の割込処理454の一つとして動作する。割込ハンドラ307で説明したとおり、遅延要求割込ハンドラ305の処理中は、第一OS301への遅延要求割込の割込レベルよりも割込レベルの高い割込を受付けることが可能である。

【0127】遅延要求割込ハンドラ305は、遅延要求キュー204から遅延要求を一つ取出す(処理531)。遅延要求が残っていないことがわかった場合(処理532)、第一OS遅延要求割込許可フラグ209をONにして(処理533)、処理を完了する。

【0128】遅延要求が取出せた場合、その要求の種類を判定し(処理534)、各々対応する処理を行う。ここまでで説明してきたOS間メッセージ通信の場合、メッセージの送信により、受信待機していたタスクを再開する要求(以下、「タスク再開要求」と呼ぶ)が取出されることになる。この場合、タスク再開要求には再開すべきタスクのタスクIDがパラメータとして付随しているので、受信待機する際に記録されたタスクIDとイベントIDの組から当該タスクIDに対応するタスクを再開するためのイベントIDを取得し、当該イベントをシグナル状態にするシステムコールを発行する(処理535)。

【0129】イベントがシグナル状態になると、当該タスクで実行中だったOS切替部利用ライブラリ310のメッセージ受信処理(処理470)が再開され、送信されたメッセージの取出しが行われることになる。遅延要求割込ハンドラ305は、遅延要求キュー204の要求が無くなるまで、要求の取出し(処理531)を繰り返す。

【0130】なお、ここで説明した遅延要求割込ハンドラ305の処理の大部分を、割込ハンドラではなく第一OS301上で動作するタスクとして構成することも可能である。すなわち、遅延要求割込ハンドラ305は、当該処理タスクに割込の発生を通知するだけで処理を終了する。当該処理タスクでは前記通知を受け次第、ここまでで説明してきた遅延要求割込ハンドラ305の処理を実行する。第一OS遅延要求割込許可フラグ209をONにする処理533が実行されるまでは、再度遅延要求割込が発生することはないので、遅延要求割込ハンドラと当該処理タスクの間で競合管理を行う必要はない。

【0131】ここまでの説明では、遅延要求割込ハンドラ305が直接遅延要求キュー204や遅延要求割込許

可フラグ209を操作するようにしている。これらのキューやフラグには遅延要求管理モジュール203もアクセスするため、競合管理が必要となる。第一OS301上のコンポーネントである遅延要求割込ハンドラ305とOS切替部201上のコンポーネントである遅延要求管理モジュール203との間で競合管理をするには、複雑な処理が必要になる。これを解決するには、遅延要求管理モジュール203に、遅延要求キュー204からの要求取出し処理や遅延要求割込許可フラグ209の操作処理を用意し、遅延要求割込ハンドラ305はこれらの処理を呼出す形とすればよい。

【0132】以上で説明した動作により、異なるOS上で動作するタスク間でのメッセージ通信機能が実現される。

【0133】第四に、遅延要求を使用するその他の処理について説明する。既に述べたとおり、遅延要求割込ハンドラ305は当該OSのシステムコールを発行できるため、OS切替部201から各OSに対する様々な処理要求の実現手段として使用できる。

【0134】まず、OS間通信の他の処理での使用を説明する。OS間メッセージ通信のメッセージキューが満杯の場合、当該メッセージキューに対する送信要求をメッセージキューにメッセージを格納する空きが出来るまで待機させる処理を実装することができる。この場合、前述のメッセージ受信待機と同様にタスク再開要求の遅延要求を使用すれば実現できる。

【0135】ただし、送信待機の場合は受信待機と異なり、複数の要求を同時に待機させてもよい。そのようにした場合、サイズの大きなメッセージがメッセージキューより取出されたときに、複数のタスク再開要求を同時に発行する必要がある。この場合、遅延要求管理モジュール203へは全ての要求を一括して追加するように依頼し、遅延要求キューに格納することが可能である。

【0136】しかし、これら複数の要求が複数のOSに対するものでありなおかつこれらのOSがいずれも遅延要求割込を発行できる状態にあった場合でも、いずれか一つのOSにしか遅延要求割込は発行できず、他のOSへの割込発生は再度遅延要求管理モジュール203が呼出されるまで遅れることになる。

【0137】次に、OS間通信モジュール206の機能の一つとして、異なるOSで動作するタスク間で競合管理を行うためのOS間セマフォを実装する場合について説明する。

【0138】OS間セマフォは、基本的にはOS間メッセージ通信の送受信処理と同様にして実現できる。すなわち、セマフォの獲得が成功した場合には獲得処理が即座にリターンし、一方獲得待ちとなった場合には処理保留でリターンする。そして、セマフォの解放処理で、セマフォ獲得待機タスクがある場合には当該タスクのタスク再開遅延要求を行えばよい。

【0139】一方、OS間セマフォの追加的な機能として、優先順位の継承を実装する場合がある。これはOS間セマフォの獲得待機をしているタスクの実行優先順位が、既に当該セマフォを獲得しているタスクの優先順位よりも高かった場合に、既に当該セマフォを獲得しているタスクの優先順位を一時的に獲得待機をしているタスクの実行優先順位まで引上げる機能である。このためには、OS間通信モジュール206から各OSに対して、タスクの優先順位変更を依頼しなければならない。この依頼は、タスクの優先順位変更という遅延要求を追加し、遅延要求割込ハンドラ305内で当該要求取出し時にタスクの優先順位変更システムコールを発行することで実現できる。

【0140】次に、OSのシステム時刻変更時の動作について説明する。既に述べたとおり、各OSは、RTC107に対して直接時刻を読み書きする代りに、RTC管理モジュール208にこれらの要求を行う。第二OS351で管理するシステム時刻が変更された場合、変更後の新しいシステム時刻をRTC107にも反映させるため、RTC管理モジュール208に設定依頼が行われる。

【0141】この場合の処理フローが、図25である。依頼が合った場合、実際にRTC107へ時刻を設定し（処理551）、遅延要求管理モジュール203にOS1に対する「時刻変更要求」の遅延要求の発行を依頼する（処理552）。遅延要求割込ハンドラ305は、図24に示したように、取出した遅延要求が「時刻変更要求」であった場合には、RTC107の時刻を取得し（処理536）、システム時刻をこの値に設定するシステムコールを発行する（処理537）。

【0142】ただし、処理537の実行中に、当該システムコール発行の結果再度RTC管理モジュール208のRTC設定処理が呼出されないように予め設定しておく必要がある。なお、RTC時刻にシステム時刻を一致させるシステムコールがあれば、処理536ならびに処理537は、一括して当該システムコールに置換えることができる。また、より簡易な処理としては、処理536を実行する代りに、処理522において、「時刻変更要求」の遅延要求に付随するパラメータに設定すべき時刻のデータを付加しておく方法も可能である。

【0143】さらに、OS状態管理モジュール207からの要求も遅延要求として実現することが可能である。図24の処理フローではその一例として、「シャットダウン要求」の遅延要求処理を含んでいる。遅延要求割込ハンドラ305は、取出した遅延要求が「シャットダウン要求」であった場合には、シャットダウンを開始するシステムコールを発行する（処理538）。シャットダウン要求の他に、あるOSの動作状態が変化した場合、たとえばOSが起動した場合、OSがシャットダウンした場合、OSが異常停止した場合など、これを通知する

ための遅延要求を他のOSに行うことができる。遅延要求割込ハンドラ305が、このようなOSの状態変化を通知する遅延要求を受取った場合、状態変化を取扱うタスクを起動する、状態変化を通知するイベントをシグナル状態に変更する、といった処理を行うことが可能である。

【0144】図5では、遅延要求キュー204、205は各OS301、351ごとに一つだけ存在するように示されているが、以上で説明したような各遅延要求の重要性のレベル毎にキューを用意して、対応するキューに分けて蓄積することが可能である。

【0145】このような遅延要求キューの構成例を図6に示す。図6の構成の場合、第一OS遅延要求キュー204は、レベル1遅延要求キュー2041、レベル2遅延要求キュー2042、レベル3遅延要求キュー2043の集合体として構成される。レベル1遅延要求キュー2041にはシャットダウン要求などのOS状態管理モジュール207からの処理要求が、レベル2遅延要求キュー2042にはRTC管理モジュール208から時刻変更要求が、レベル3遅延要求キュー2043にはタスク再開要求などのOS間通信モジュール206からの処理要求が蓄積される。

【0146】ここで、遅延要求割込ハンドラ305が、レベルの小さいキューから順に遅延要求を取出すことにより、各要求はレベル1、レベル2、レベル3の順で優先して処理されることになる。なお、遅延要求管理モジュール203の中に遅延要求の取出し処理を実装する場合には、当該処理の中でレベルの小さいキューから順に遅延要求を取出すことになる。図6では、遅延要求キューを三つのキューで構成されたとしたが、キューの数を変更し各キューに優先順位を付けることも可能である。また、各キュー毎に、遅延要求割込の割込レベルを異なるものとしたり、遅延要求割込を発生させることができるOSの動作優先順位を変更したりすることも可能である。

【0147】以上のように、OS切替部201から各OS301、351に対する様々な処理要求の実現手段として、遅延要求が使用される。しかし、処理内容によっては遅延要求が適さず、OS切替部201から呼出されるコールバックルーチンで処理する必要がある。

【0148】遅延要求とコールバックルーチンのいずれを使うかの判断基準の例を図30に示す。図30で示した判断基準の場合、処理によりスケジューリングが発生する可能性がある場合や処理時間が長く処理中にOS切替や割込発生を許可したい場合に遅延要求を使用し、即座に実行が必要な処理や当該OSの割込処理が正常に行われていない場合の処理、あるいは処理によりスケジューリングが発生しない短時間の処理にコールバックルーチンを使用する。例えば、OS状態管理モジュール207から当該OSを強制停止させる処理の場合、当該OS

が正常に動作していない場合に使われる可能性があるため、遅延要求を使用せずにコールバックルーチンを使用する必要がある。ただし、長時間の処理をコールバックルーチンで処理することは可能であり、また処理結果でスケジューリングが発生しない短い処理を遅延要求で処理することも可能である。

【0149】以上で説明した動作により、遅延要求を使用する各処理が実現される。

【0150】第五に、OS状態管理モジュール207の各OSに対する監視タイマ機能(Watch Dog Timer。以下、「WDT」と呼ぶ)の動作について説明する。

【0151】図7にWDT機能に関するソフトウェアの機能ブロック図を示す。

【0152】OS状態管理モジュール207は、第一OS301用のWDTカウンタ217、および第二OS351用のWDTカウンタ218を管理している。これらのカウンタ217、218は当該OS301、351に対するWDT機能を開始したときに0にクリアされる。また、タスクからの要求があった場合にも、当該タスクが動作しているOS用のWDTカウンタが0にクリアされる。タスクからの0クリア要求は、当該タスクが動作していることを通知する意味を持ち、以下、これを「生存通知」と呼ぶ。共通割込ハンドラ212は、一定周期で発生するタイマ装置1083からの割込発生を受け、OS状態管理モジュール207を呼出す。

【0153】OS状態管理モジュール207のタイマ割込処理(処理560)の処理フローを図26に示す。

【0154】処理では、まず、第一OS301および第二OS351用のWDTカウンタに値を1だけ加算する(処理561、処理562)。そして第一OS用WDTカウンタ217が予め指定された第一OS301のタイムアウト時間に相当するカウント値よりも大きいと判断された場合(処理563)、第一OS301のコールバックルーチン306を呼出して第一OSが管理する入出力装置からの割込発生を抑止させ(処理564)、以後OSコンテキスト切替モジュール202の処理で第一OSへ切替えることを禁止する(処理565)。第二OSに対しても同様の処理を行う(処理566～568)。

【0155】ここで説明した処理により、OSの動作の正常性を確認する機能が提供される。生存通知を行うタスクの優先順位を高く設定しておけば、OS本体や優先順位の非常に高いタスク(たとえば当該OS上の統括処理を行うタスク)が動作できなくなった場合に限り、当該OSが不正常だとみなされる。

【0156】一方、生存通知を行うタスクの優先順位を低く設定しておけば、当該優先順位以上の優先順位を持つタスクのいずれか一つに対してでも処理時間の割当が行われなかった場合に、当該OSの動作を不正常として検出することになる。生存通知を行うタスクの優先順位

は、当該OS上で動作する各タスクの重要性と、当該OSの状態管理や当該OS上の統括処理を行うタスクの処理内容を勘案して決定することになる。

【0157】本実施例では、WDTで異常を検出した場合、当該OSの動作を即座に停止するものとしている。

【0158】一方、異常を検出した場合に、まず正規のシャットダウン手順での停止を当該OSに要求し、一定時間以内にシャットダウンの完了通知がなかった場合には、強制的に停止することも可能である。このようにするには、WDTタイムアウト時間を越えた場合の処理564ないし567の代りに当該OSに対するシャットダウン要求の遅延要求を発行し、OSへの切替禁止（処理565ないし処理568）をしない。その後のタイマ割込処理においても、当該OSのWDTカウンタ加算561ないし562を継続する。

【0159】ここで当該OSからシャットダウン完了の通知があった場合はタイマの加算を行わないようにするが、当該通知がないまま加算を継続して第二のタイムアウト時間を越えた場合、当該OSの割込マスクを行うコールバックルーチン呼出（図26の処理564ないし処理567）、および当該OSへの切替禁止（処理565ないし処理568）を行えばよい。なお、ここで説明した一定時間以内にシャットダウンの完了通知がなかった場合に当該OSを強制停止するという手順については、WDTのタイムアウト時以外にも、各OSに対するシャットダウン要求を行う全ての場合に共通に適用することが可能である。

【0160】さらに、異常を検出して当該OSを停止させた後に、自動的に当該OSの起動処理を呼出し、当該OSを再起動する手順を追加することも可能である。この場合、当該OS上タスクの不良論理などによる異常検出・再起動の繰返しを防ぐために、OS状態管理モジュール207内に各OSの再起動回数を記録する変数を用意して、再起動回数が指定回数以上になった場合は自動再起動を抑止する機能を追加することが可能である。

【0161】この再起動回数の記録は、指定時間ごとに、あるいは最後の再起動が行われてから指定時間が経過した後にクリアする機能を追加することが可能である。また、再起動を要求してから指定した時間内に再度異常を検出した場合に、正常な起動が出来ないものとみなして自動再起動を抑止する機能を追加することが可能である。なお、ここで説明した自動再起動の処理については、WDTのタイムアウト時以外に、各OSの異常終了検出時にも共通に適用することが可能である。

【0162】以上で説明した処理により、当該計算機システムの信頼性を向上させるためのOS動作管理モジュール、特にWDT機能の動作が実現される。

【0163】第六に、OS間リモート手続き呼出機能（Remote Procedure Call。以下「RPC機能」と呼ぶ）の動作について説明する。

【0164】図8にOS間RPC機能に関するソフトウェアの構成図を示す。OS間RPC機能を使用する第一OS301上のタスクはRPC利用ライブラリ311に用意された処理関数を呼出す。RPC利用ライブラリ311は、OS間通信モジュール206のOS間メッセージ通信機能を使用して、第二OS351上にあるRPCサーバ354に処理を依頼する。このとき使用されるメッセージキュー219は第二OS351のRPCサーバ354に対する処理要求を送るための専用のもので、そのメッセージキューIDはRPC利用ライブラリ311が予め知っているものとする。

【0165】RPCサーバ354は、実際にはRPC管理タスク3541とRPC実行タスク3542の組合わせで構成される。RPC実行タスク3542は、要求された関数（手続き）を実行する。この関数は通常のタスク環境で実行されているので、システムコール358を発行することが可能である。また、要求関数として、システムコールそのものを指定することも可能である。RPC管理タスク3541は要求された処理が完了した後、当該処理要求を行ったタスクA302専用のRPC結果通知メッセージキュー220を使用して、処理結果を通知する。

【0166】RPC利用ライブラリ311の初期化処理（処理580）の処理フローを図27に示す。この初期化処理は、各タスクがRPCを初めて利用するとき一度だけ呼出される。この処理を呼出したタスクが、その後のRPC処理要求を行った場合に、処理結果を受取るためのRPC結果通知メッセージキュー220を作成する（処理581）。

【0167】RPC利用ライブラリ311のRPC実行処理（処理590）の処理フローを図28に示す。タスクは、実行する関数名とパラメータを渡して処理590を呼出す。処理590では、第二OS351のRPC要求メッセージキュー219に対して、要求元から指定されたままの関数名とパラメータ、および初期化処理580で作成したRPC結果通知メッセージキュー220のメッセージキューIDを、一つの要求メッセージとして送信する（処理591）。その後、RPC結果通知メッセージキュー220に対して結果を通知するメッセージが送られてくるのを受信待機する（処理592）。

【0168】第二OS351のRPC管理タスク3541の処理フローを図29に示す。管理タスクは常時RPC要求メッセージキュー219の受信待機をしておりメッセージが到着次第これを取り出す（処理601）。要求メッセージを取り出すと、RPC実行タスク3542を起動し（処理602）、要求された実行関数名およびパラメータをRPC実行タスク3542に渡して実行開始を指示する（処理603）。そしてRPC実行タスク3542からの実行完了メッセージが送られてくるのを待機する（処理604）。なお、ここでいうメッセージは、

第二OS 351の機能として提供される同期機能あるいは通信機能のいずれかを使用して実現する。

【0169】PRC実行タスク3542は、関数名から当該関数のアドレスを返す第二OS 351のシステムコールを呼出すことにより要求された関数のアドレスを解決し、当該関数を実行する。実行が完了した場合、関数のリターン値を含む終了メッセージをRPC管理タスク3541に送る。ここで、要求された処理やパラメータの正当性チェックは行わないため、処理が完了しなかったり、PRC実行タスク3542が停止してしまったりする場合がある。

【0170】第二OS 351のRPC管理タスク3541は実行完了メッセージの待機処理604において、タイムアウトを設定しており、前記のようにPRC実行タスク3542が正常に終了しない場合にはタイムアウトとなる（処理605）。この場合は、PRC実行タスク3542を強制的に停止し（処理608）、処理要求元タスクのRPC結果通知メッセージキュー220に対してタイムアウトエラーの結果メッセージを送信する。

【0171】一方、PRC実行タスク3542が正常に処理を終了して、終了メッセージが送られてきた場合、PRC実行タスク3542を停止し（処理606）、終了メッセージに含まれるリターン値を含む結果メッセージを、処理要求元タスクのRPC結果通知メッセージキュー220に対して送信する（処理607）。

【0172】RPC利用ライブラリ311のRPC実行処理（処理590）は、結果通知メッセージの到着で動作を再開し、その結果とともに、処理呼出元にリターンする（処理593）。ここで、あるタスクがRPC実行処理590を呼出すと、この処理が終了するまで当該タスクは実行を中断することになる。

【0173】RPC管理タスク3541は予め決められた処理を実行するだけであり要求された処理の内容にかかわらず異常終了することではなく、またRPC実行タスク3542に対するタイムアウトが設定されているため、必ず何らかの処理結果のメッセージが送られる。すなわち、処理590に相当する関数内で指定したRPC処理が完結するかタイムアウトするかでリターンしてくることが期待でき、異なるOS間で通信する処理や、通信に伴うエラー処理などを意識することなく、他のOS上での処理が実行できる。なお、ここではタイムアウト時間をRPCサーバが保持している固定の値としたが、個々のRPC処理要求毎にタイムアウト時間を指定するようにしてもよい。

【0174】ここで説明したRPCサーバの構成では、RPC実行タスク3542でエラーが発生して強制停止した場合にはRPC管理タスク3541内でのタイムアウト発生まで待つものとしたが、タスクの停止を通知する機能が当該OSに備わっている場合には、この通知を検出して、即座にエラーの通知を行う構成も可能であ

る。

【0175】また、RPCサーバの構成では、関数アドレスの解決はRPC実行タスク3542で行うものとしたが、これをRPC管理タスク3541で実行し、RPC実行タスク3542には解決済のアドレスを渡す構成も可能である。さらに、一旦解決した関数アドレスをRPC管理タスク3541内に置いたテーブルに記録し、同一の関数を再度実行する場合には当該テーブルからアドレスを取得することによって、アドレスの解決処理のオーバーヘッドを小さくできる。また、システムコールによる関数アドレスの解決の代りに、予めRPC管理タスク3541のビルド時にアドレスを解決して、前記記録テーブルに保持しておく方法も可能である。

【0176】RPCサーバは、RPC実行タスク3542を一つとしたが、複数のRPC要求を同時に実行するために、RPC実行タスク3542を複数同時に起動する構成も考えられる。ただし、この場合には、起動したRPC実行タスク3542全てに対して実行完了メッセージをタイムアウト付で待機することが必要となる。例えば、起動したRPC実行タスク3542と同数の管理サブタスクを別に起動し、各々一つのRPC実行タスク3542からの実行完了メッセージを待機するといった処理が必要になる。逆に、簡易な構成として、RPC実行タスク3542の処理をRPC管理タスク3541の一部とする方法も可能である。この場合、要求関数の実行で例外が発生した場合に、当該OSが提供するエラー回復手段（シグナル機構や構造化例外処理などの手段）を使用してタスクの動作を継続させ、エラー発生を処理要求元タスクのRPC結果通知メッセージキュー220に対して送信する構成にしなければならない。

【0177】以上で説明した処理により、異なるOS上の処理を用意に呼出すためのOS間RPC機能の動作が実現される。

【0178】以上のようにして複数個のオペレーティングシステム間で処理要求を通信するのであるが、オペレーティングシステムが処理要求を割込み処理するようにしているので、オペレーティングシステムに通信用のハンドラ機能を一つ追加するだけでよくオペレーティングシステム間の処理要求の通信をオペレーティングシステムの簡単な改造で行える。

【0179】なお、上述の実施例においてはハード構成で説明しているが、プログラムの処理手順としてCD-ROMなどの記憶媒体に収納し、コンピュータなどの演算処理装置を用いて本発明を実施できることは明らかなことである。

【0180】

【発明の効果】本発明によれば、オペレーティングシステムが処理要求を割込み処理するようにしているので、オペレーティングシステムに通信用のハンドラ機能を一つ追加するだけでよくオペレーティングシステム間の処

理要求の通信をオペレーティングシステムの簡単な改造で行える。

【図面の簡単な説明】

【図1】本発明の一実施例の構成を示す図である。

【図2】計算機システムのハードウェア構成を示す図である。

【図3】割込レベルマスク機能を装備する場合のステータスレジスタを示す図である。

【図4】個別割込マスク機能を装備する場合のステータスレジスタを示す図である。

【図5】ソフトウェア構成の詳細を示す機能ブロック図である。

【図6】遅延要求キューの構成例を示す図である。

【図7】ソフトウェア構成の詳細を示す機能ブロック図である。

【図8】ソフトウェア構成の詳細を示す機能ブロック図である。

【図9】タスク管理ブロックの内部構造の例を示す図である。

【図10】タスク管理ブロックの管理例を示す図である。

【図11】スケジューラの処理フローを示す図である。

【図12】OS間優先順位管理モジュールの処理フローを示す図である。

【図13】OSコンテキスト切替モジュールの処理フローを示す図である。

【図14】共通割込ハンドラの処理フローを示す図である。

【図15】割込割当テーブルの構造の例を示す図である。

【図16】割込ハンドラの処理フローを示す図である。

【図17】OS間メッセージ通信機能の動作を示す図である。

【図18】OS切替部利用ライブラリのメッセージ受信処理の処理フローを示す図である。

【図19】OS間通信モジュールのメッセージ受信処理の処理フローを示す図である。

【図20】OS間通信モジュールのメッセージ送信処理の処理フローを示す図である。

【図21】遅延要求管理モジュールの処理フローを示す第一の図である。

【図22】遅延要求管理モジュールの処理フローを示す第二の図である。

【図23】遅延要求管理モジュールの処理フローを示す第三の図である。

【図24】遅延要求割込ハンドラの処理フローを示す図である。

【図25】RTC管理モジュールのRTC設定処理の処理フローを示す図である。

【図26】OS状態管理モジュールのタイマ割込処理の

処理フローを示す図である。

【図27】RPC利用ライブラリの初期化処理の処理フローを示す図である。

【図28】RPC利用ライブラリのRPC実行処理の処理フローを示す図である。

【図29】RPC管理タスクの処理フローを示す図である。

【図30】遅延要求とコールバックを選択する基準の例を示す図である。

【図31】OS間通信機能の従来技術を示す構成図である。

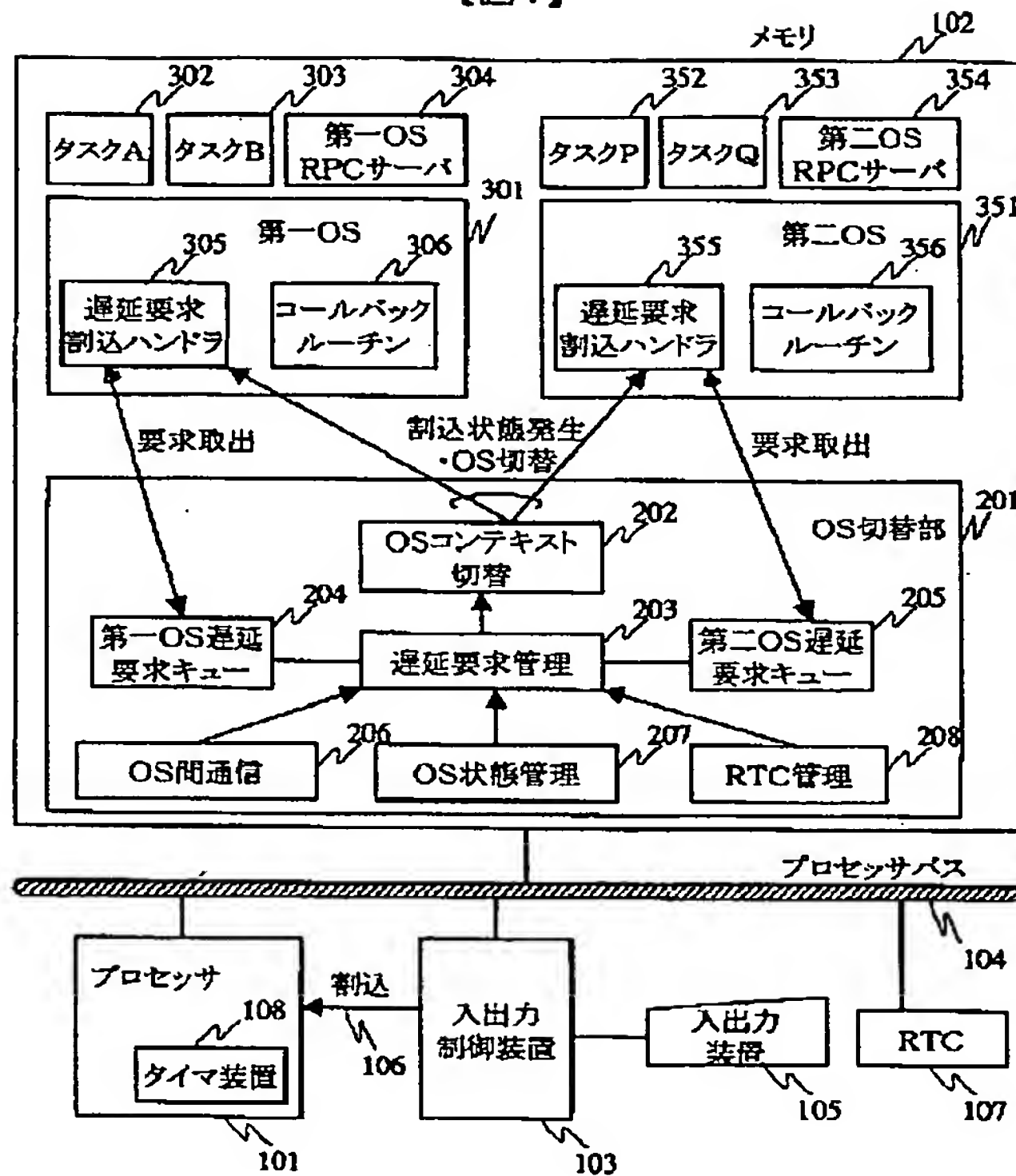
【図32】OS間通信機能の従来技術の他の例を示す構成図である。

【符号の説明】

101…プロセッサ、102…メモリ、103…入出力制御装置、104…プロセッサバス、105…入出力装置、106…割込信号線、107…RTC、108…タイマ装置、121…CPU、122…キャッシュメモリ、123…汎用レジスタ、124…プログラムカウンタ、125…ステータスレジスタ、126…割込コントローラ、127…退避プログラムカウンタ、128…退避ステータスレジスタ、129…割込アドレスレジスタ、130…割込要因レジスタ、131…データバス、132…アドレスバス、141…割込ブロックビット、142…割込マスクレベルフィールド、143…実行状態レジスタ、144…割込マスクレジスタ、145～148…割込マスクビット、149…特権モードビット、201…オペレーティングシステム切替プログラム、202…OSコンテキスト切替モジュール、203…遅延要求管理モジュール、204…第一OS遅延要求キュー、205…第二OS遅延要求キュー、206…OS間通信モジュール、207…OS状態管理モジュール、208…RTC管理モジュール、209…第一OS遅延要求割込許可フラグ、210…第二OS遅延要求割込許可フラグ、211…OS間優先順位管理モジュール、212…共通割込ハンドラ、217…第一OS用WDTカウンタ、218…第二OS用WDTカウンタ、219…RPC要求メッセージキュー、220…RPC結果通知メッセージキュー、231…割込割当テーブル、241・242…メッセージキュー、301…第一OS、351…第二OS、302・303・352・353…タスク、304・354…RPCサーバ、3541…RPC管理タスク、3542…RPC実行タスク、305・355…遅延要求割込ハンドラ、306・356…コールバックルーチン、307・357…割込ハンドラ、308・358…システムコール、309・359…スケジューラ、310・360…OS切替部利用ライブラリ、311・361…RPC利用ライブラリ、321…タスク管理ブロック、322・323…実行可能タスクキュー、324…実行中断タスクキュー

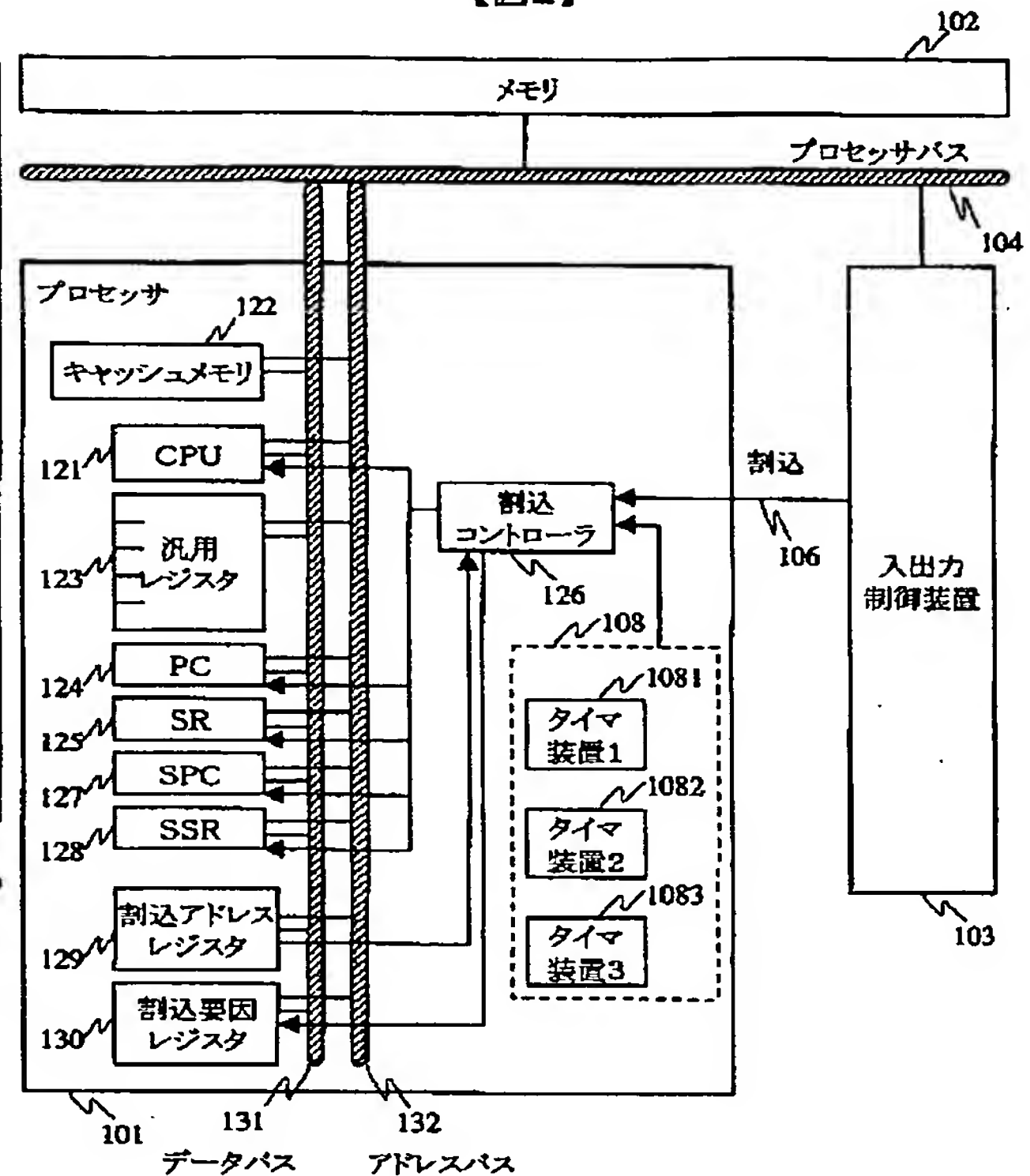
【図1】

【図1】



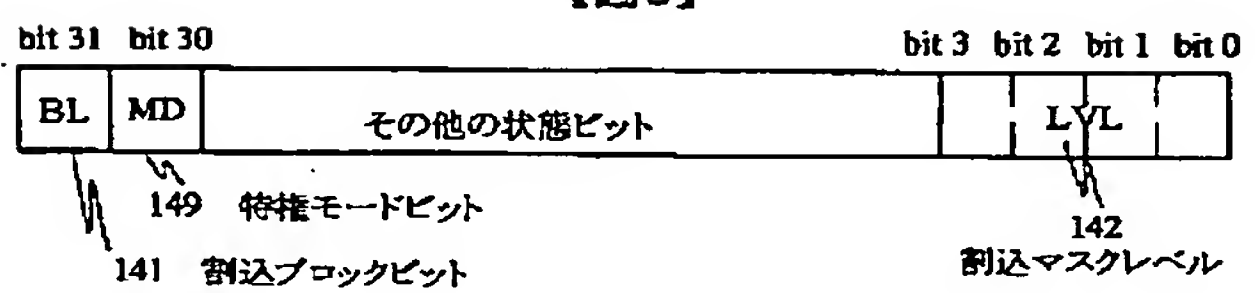
【図2】

【図2】



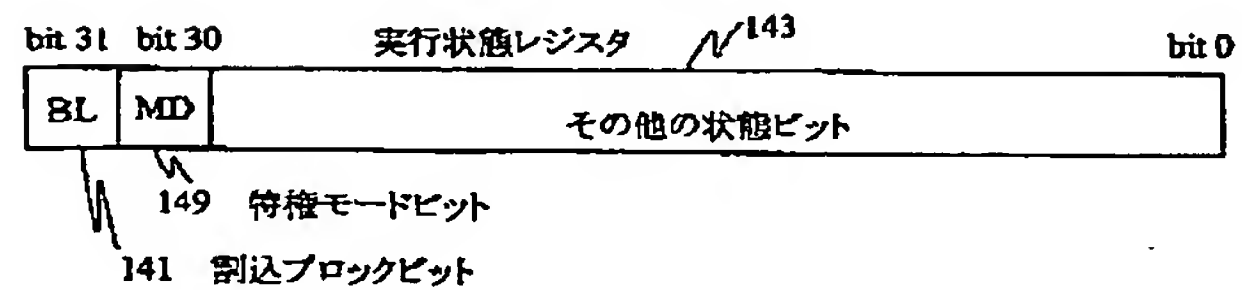
【図3】

【図3】



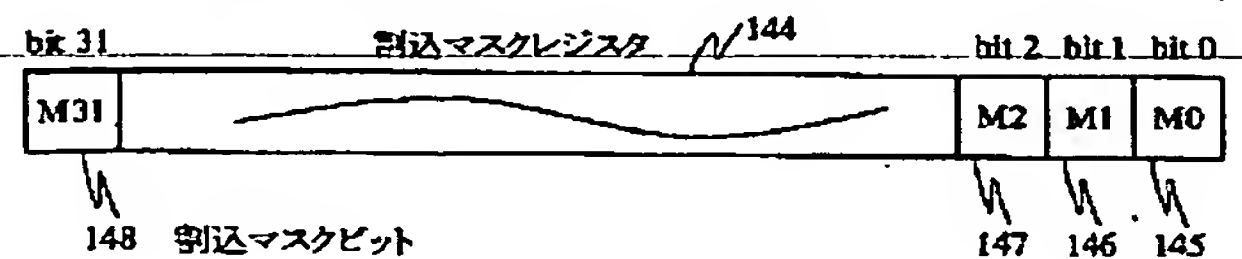
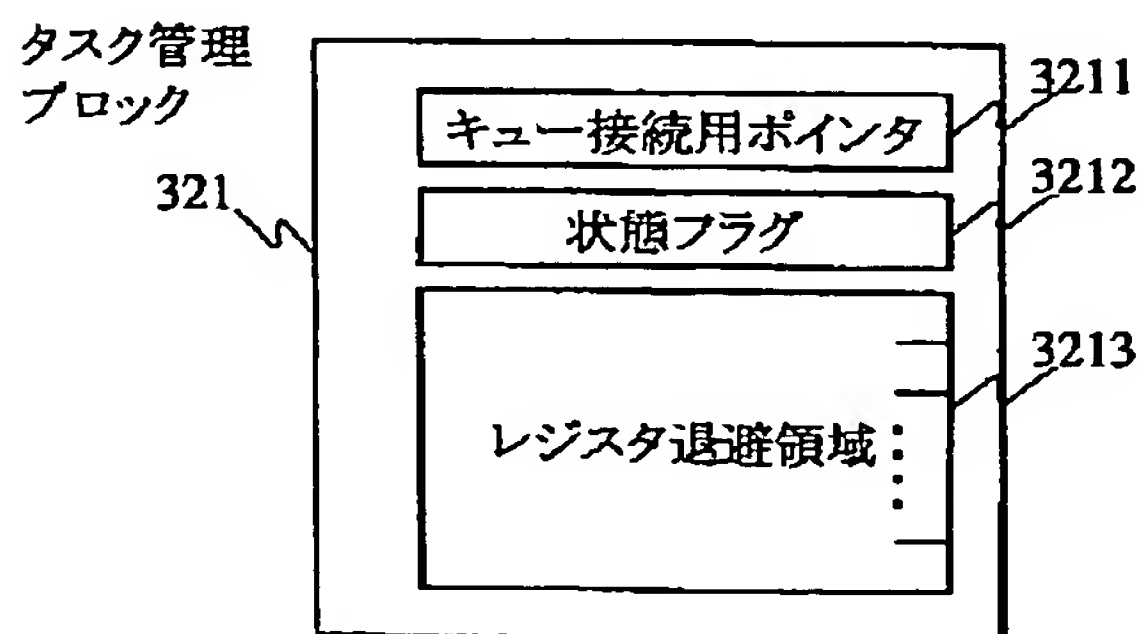
【図4】

【図4】



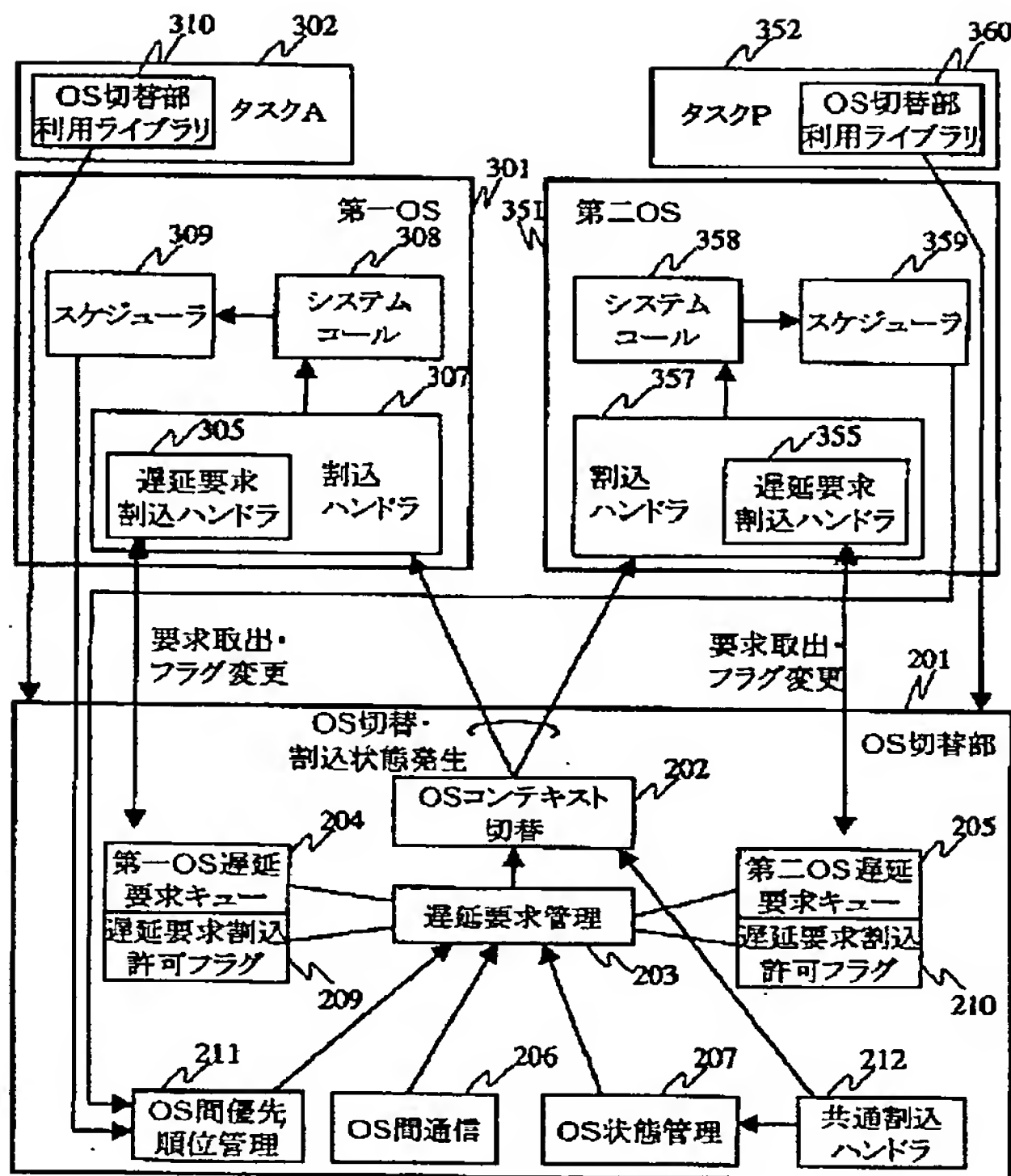
【図9】

【図9】



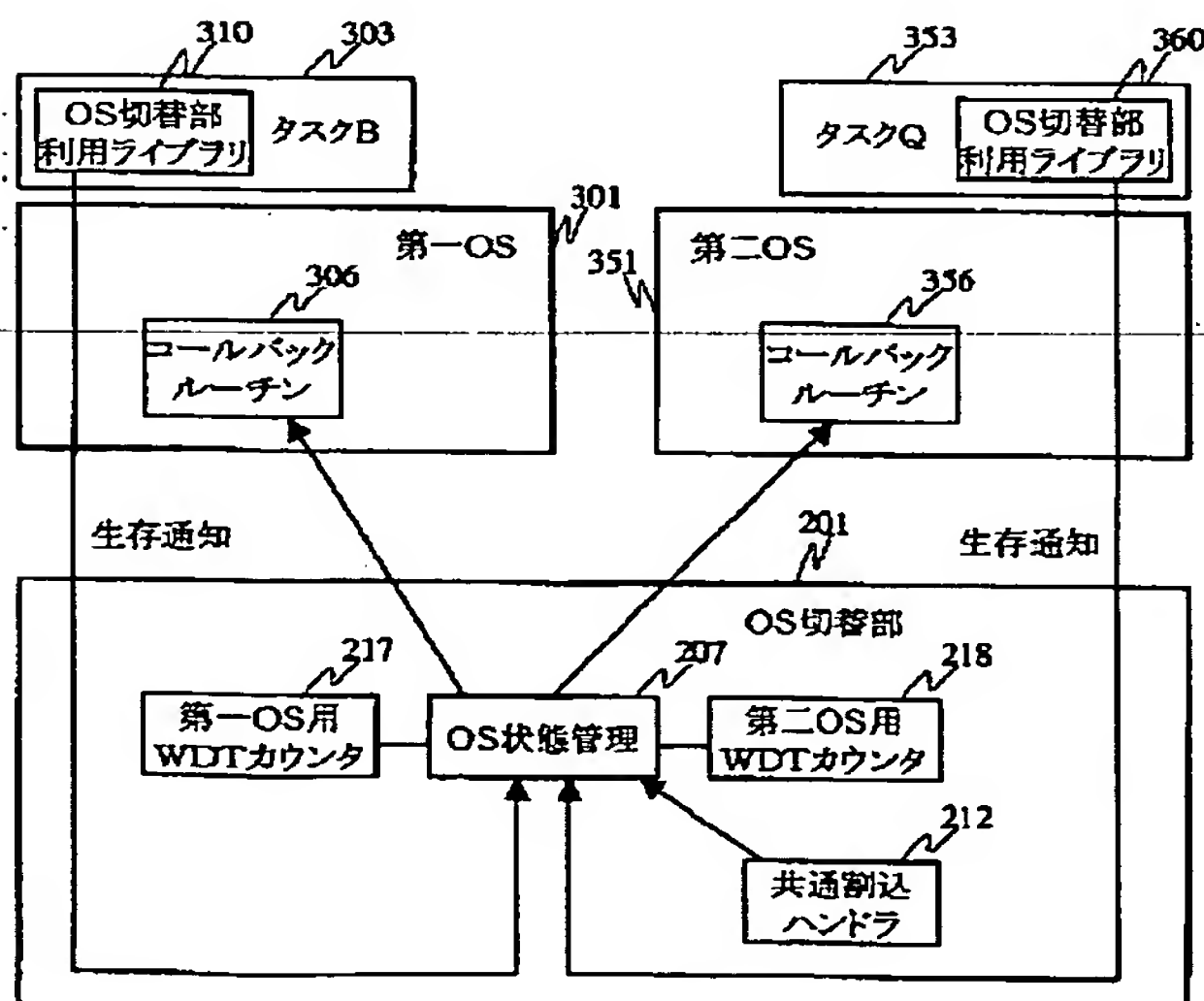
【図5】

【図5】



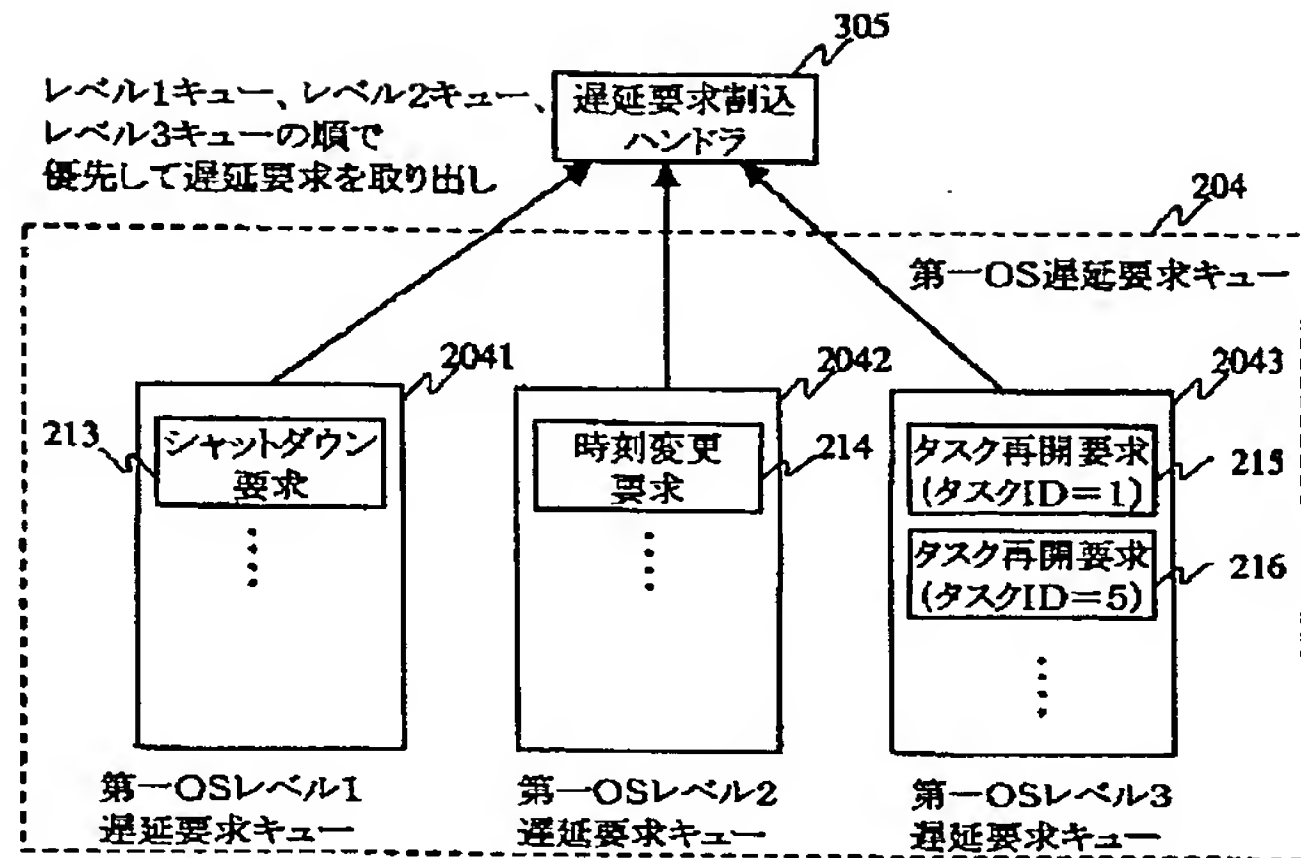
【図7】

【図7】



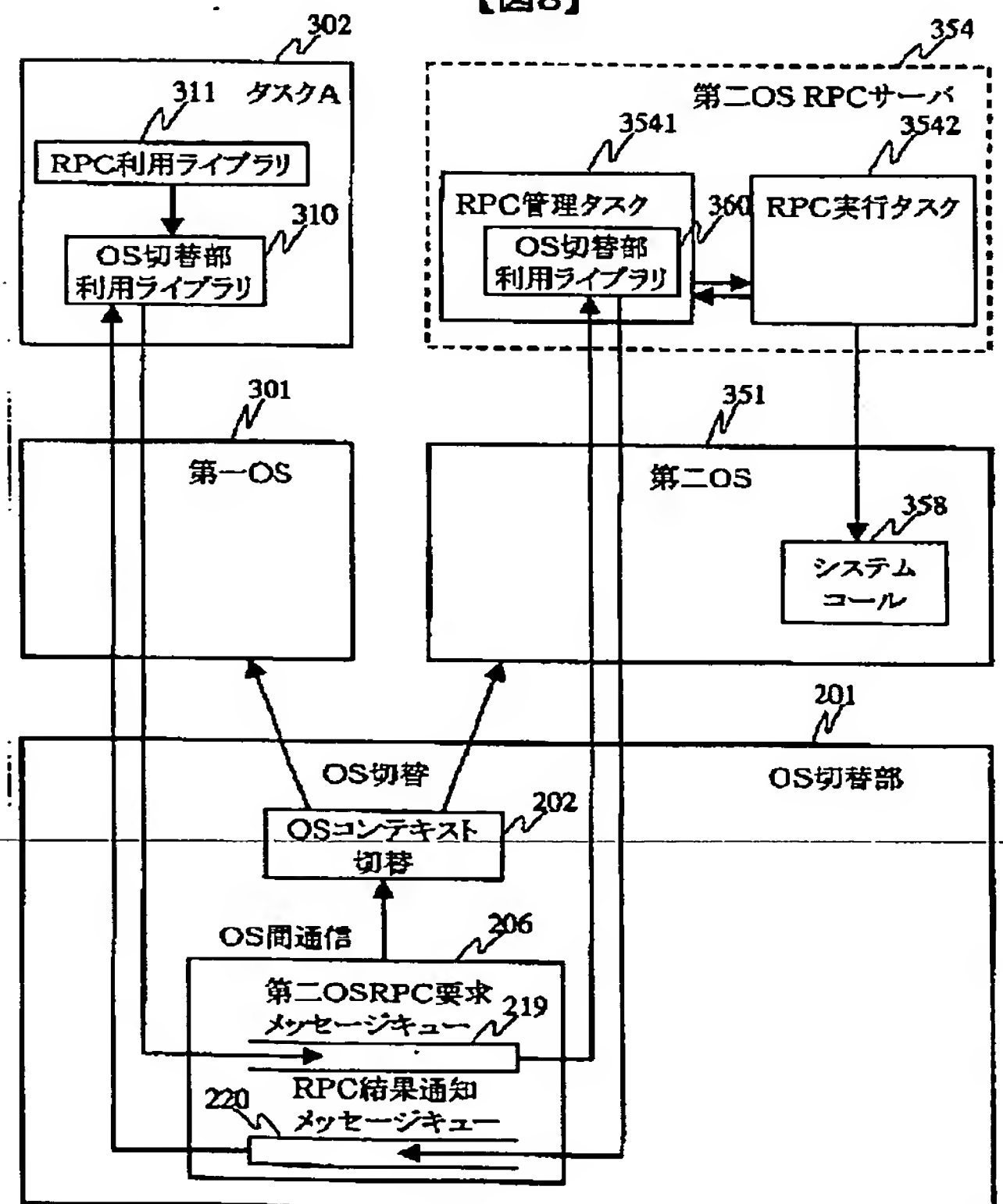
【図6】

【図6】



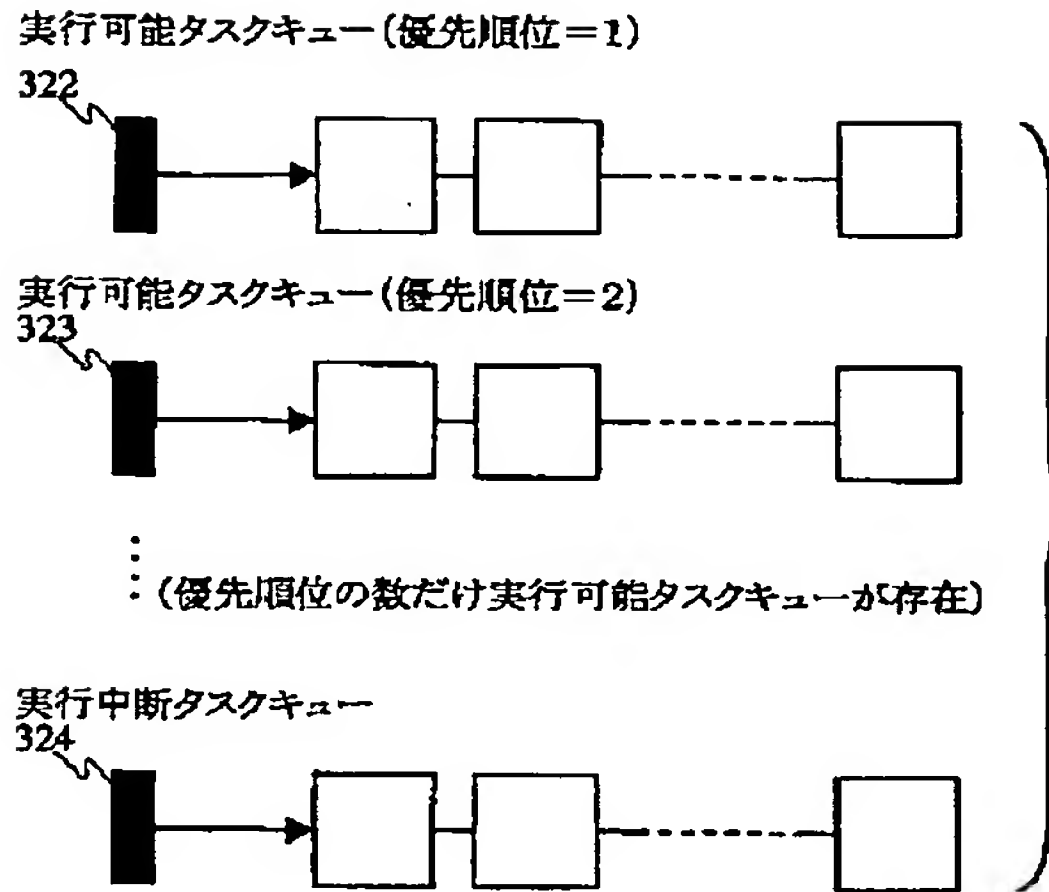
【図8】

【図8】



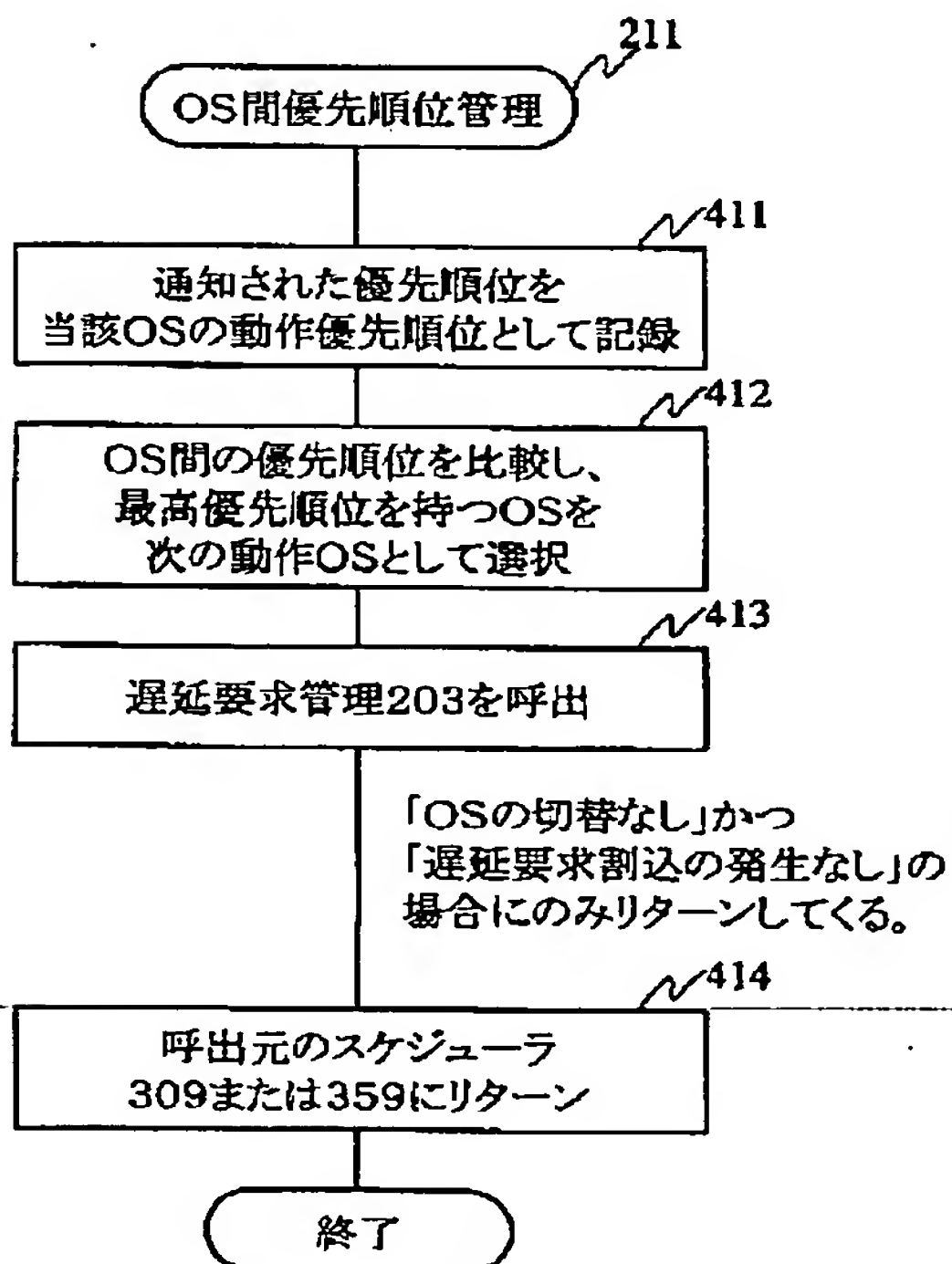
【図10】

【図10】



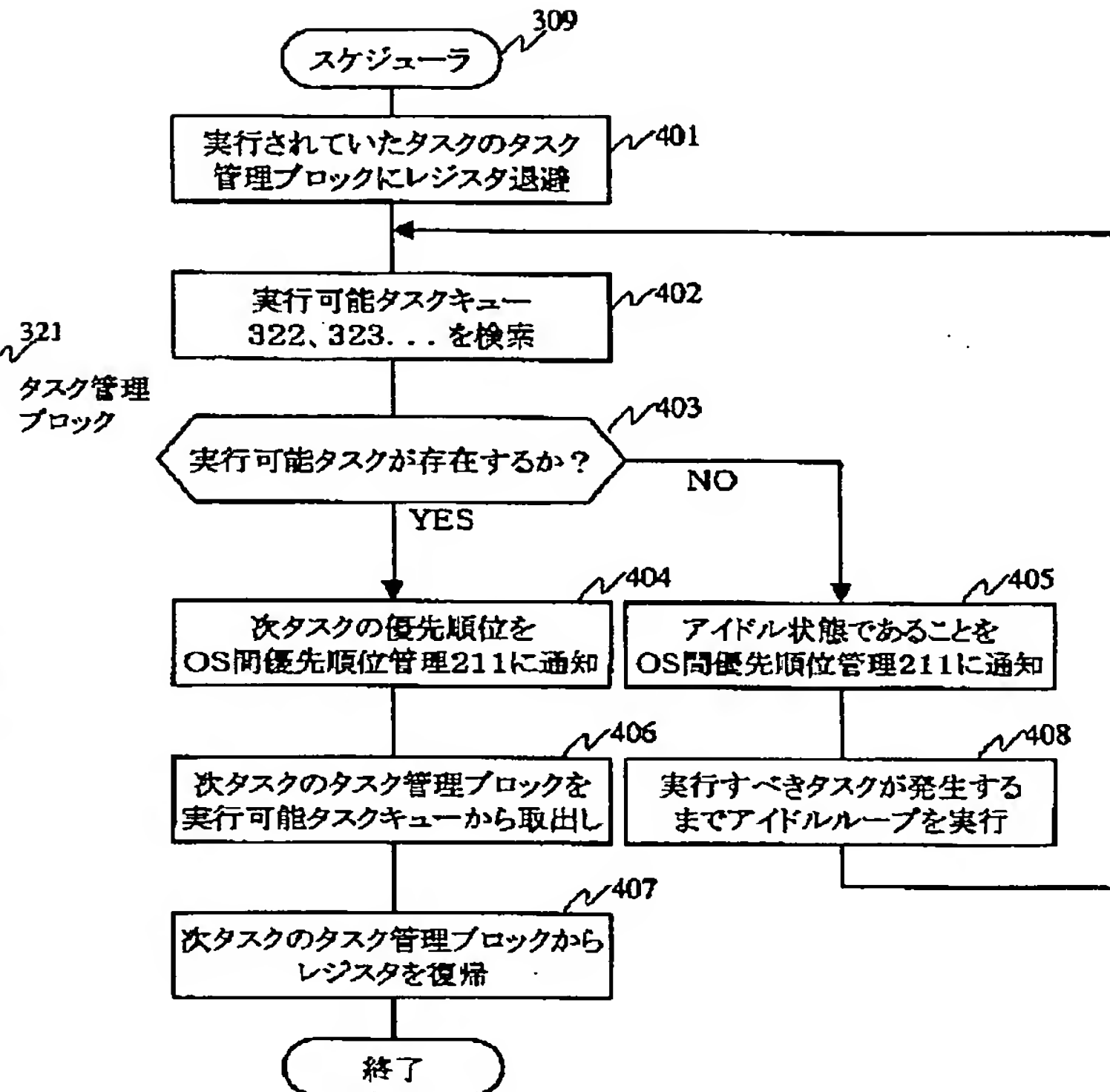
【図12】

【図12】



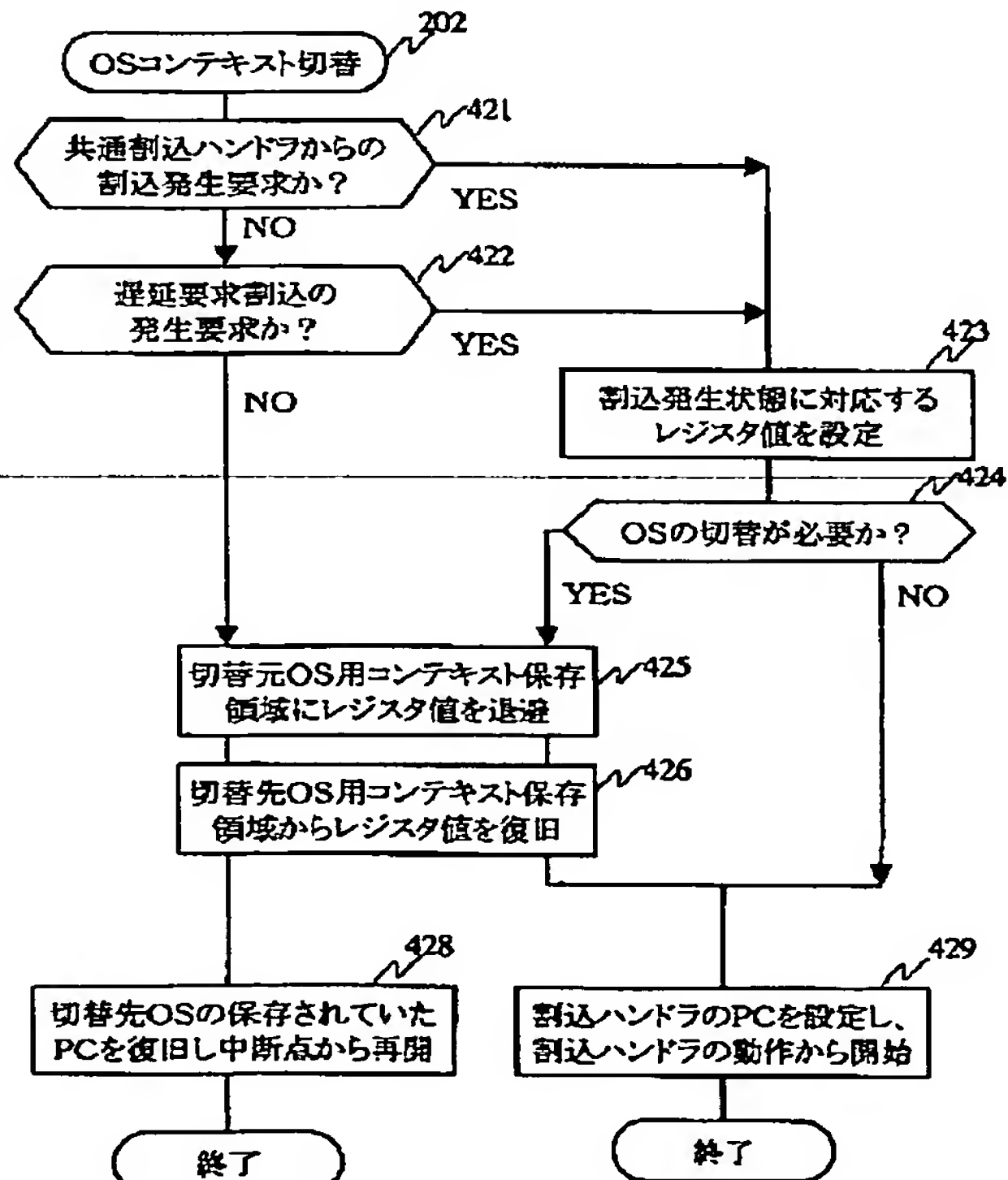
【図11】

【図11】



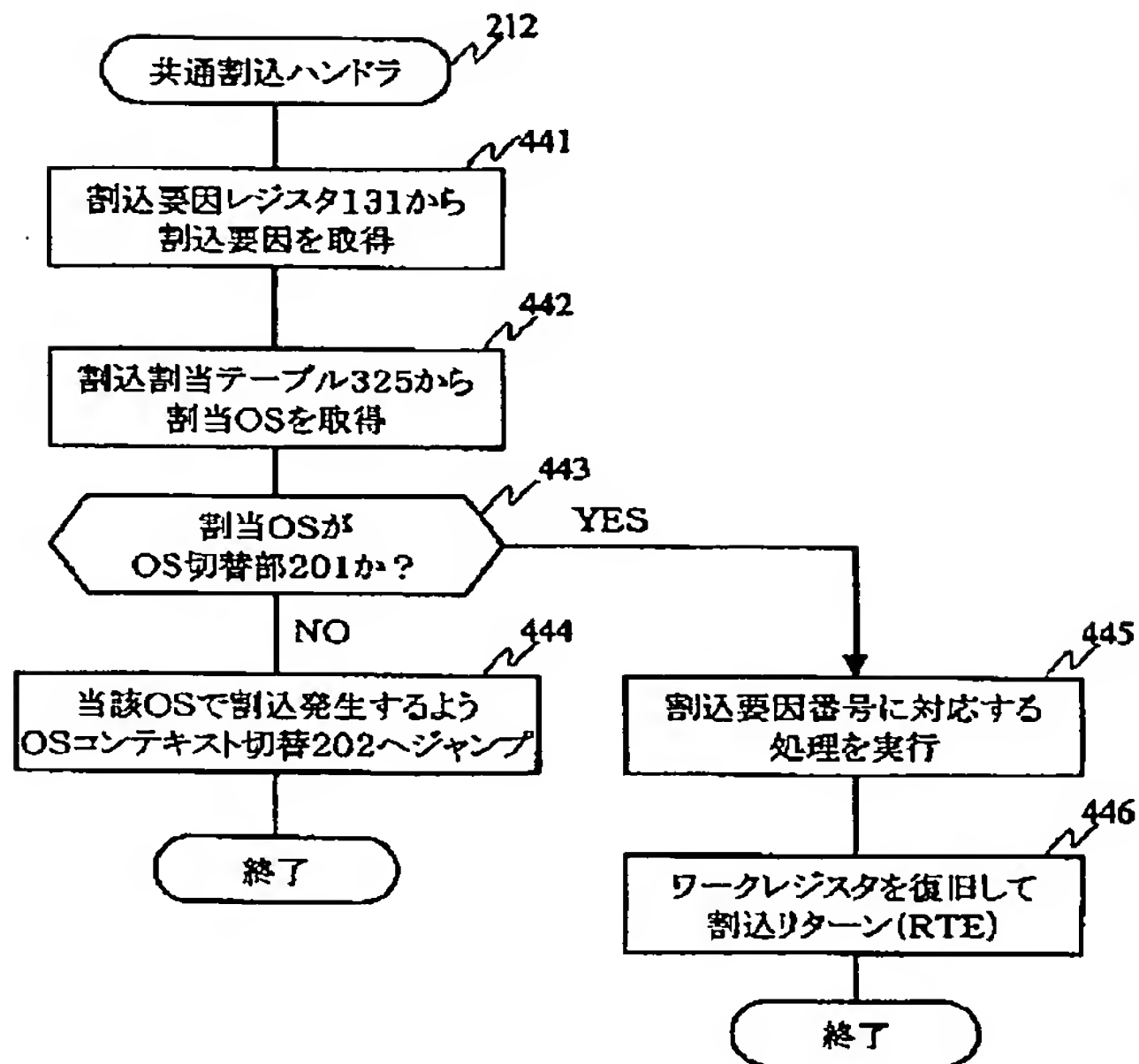
【図13】

【図13】



【図14】

【図14】



【図15】

【図15】

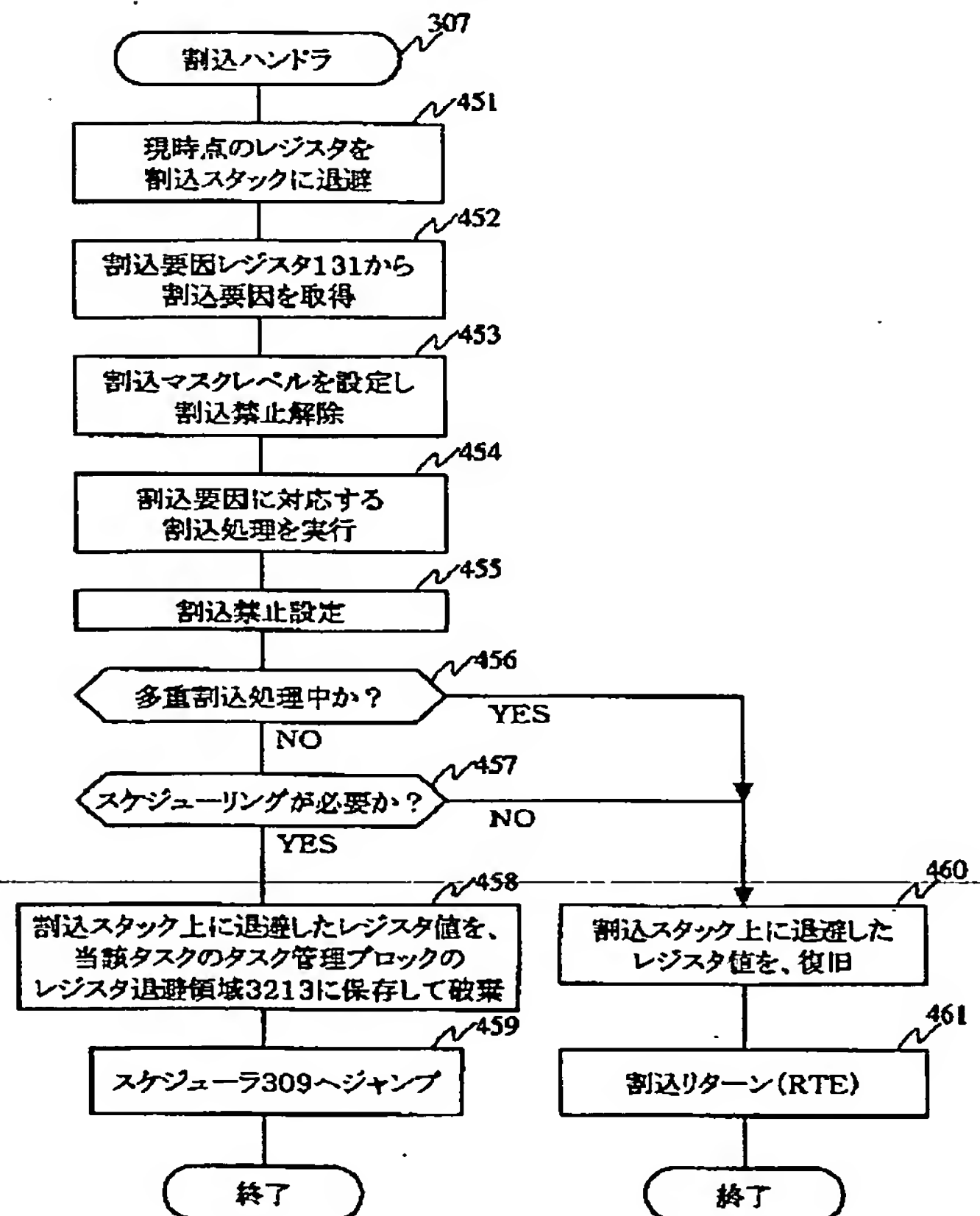
割込割当テーブル 231

割込要因レジスタ値	割当OS
0x200	第一OS
0x220	第二OS
0x240	OS切替部
0x260	第一OS
⋮	⋮

(タイマ装置1 1081からの割込)
(タイマ装置2 1082からの割込)
(タイマ装置3 1083からの割込)

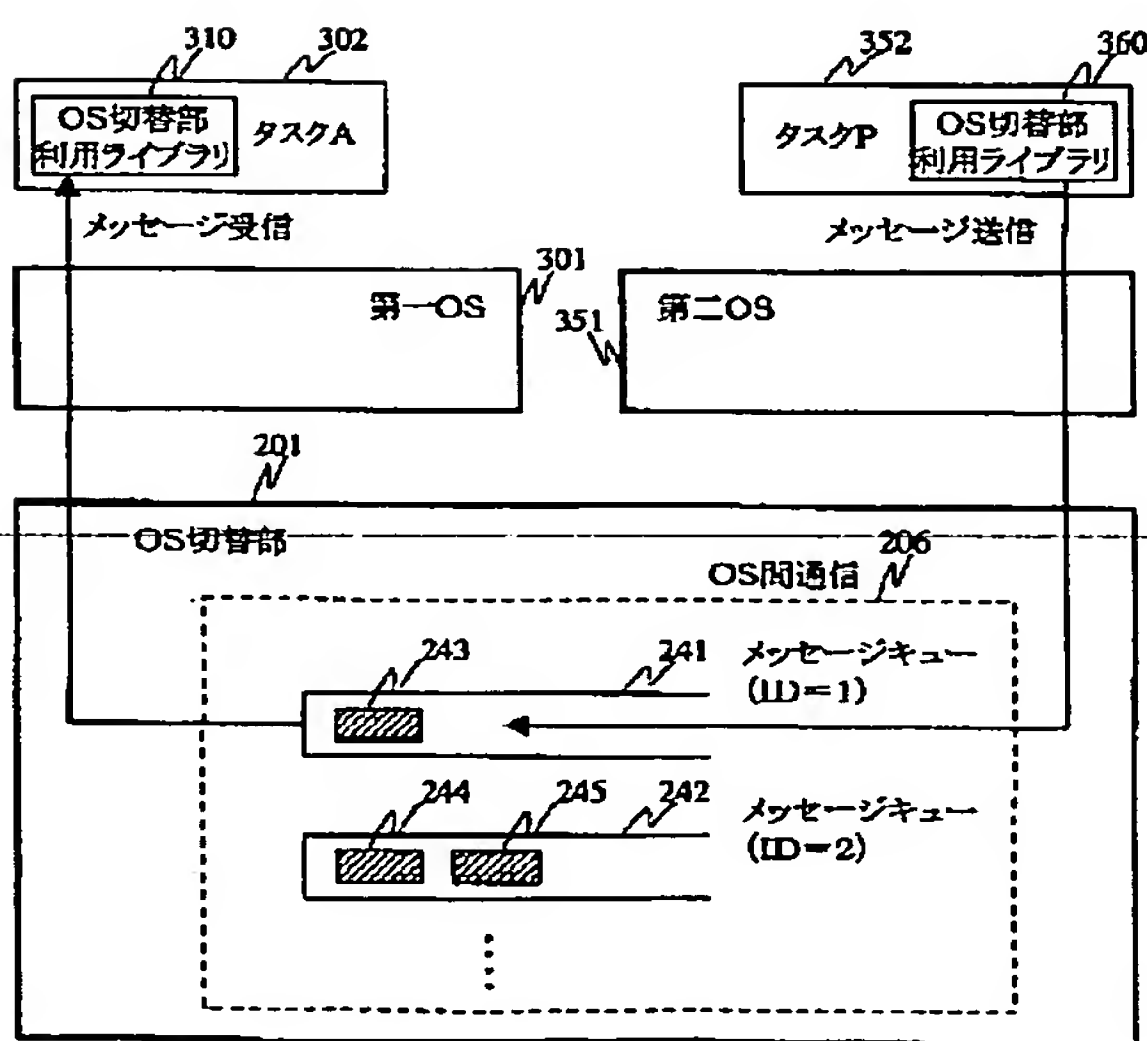
【図16】

【図16】

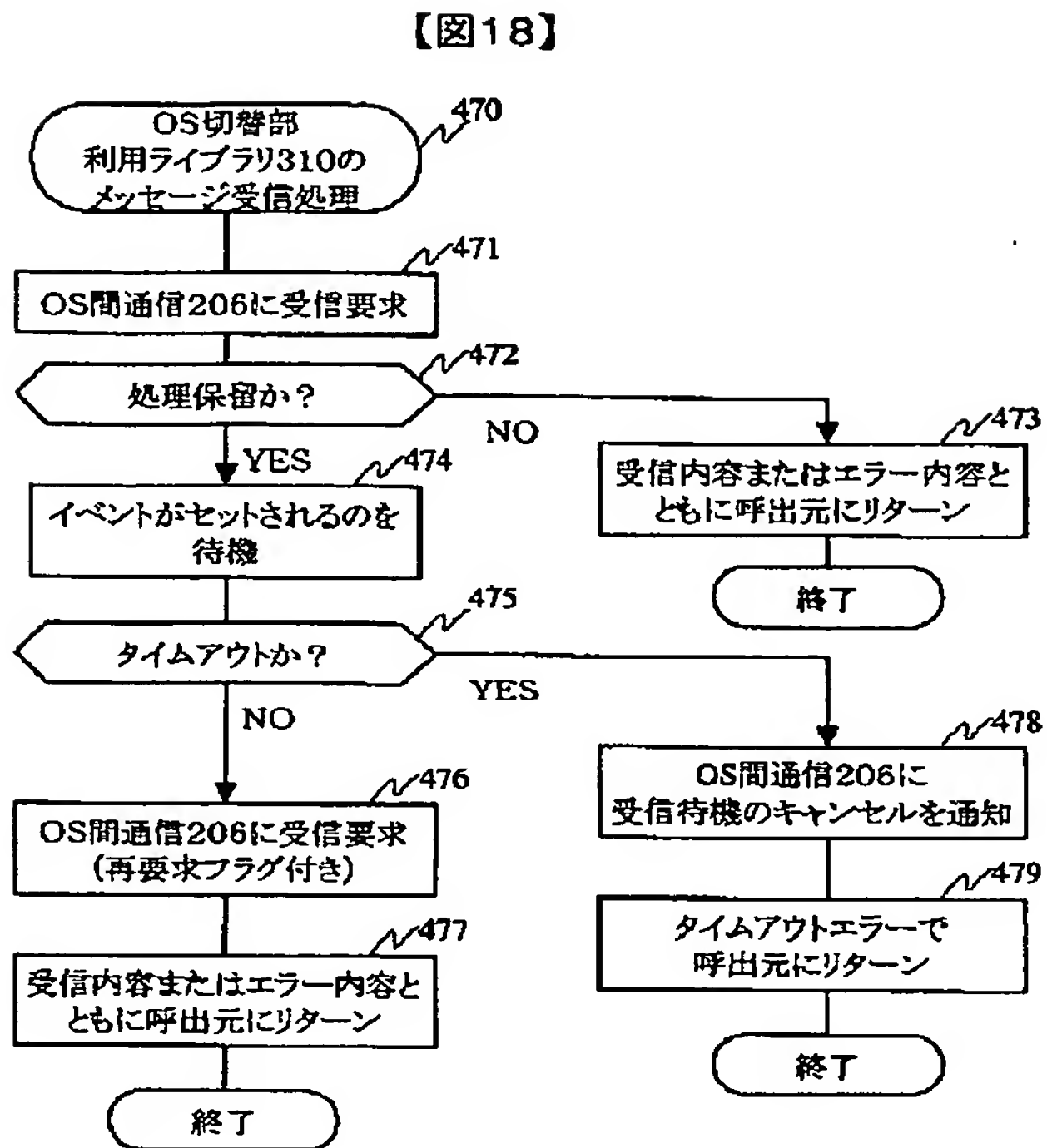


【図17】

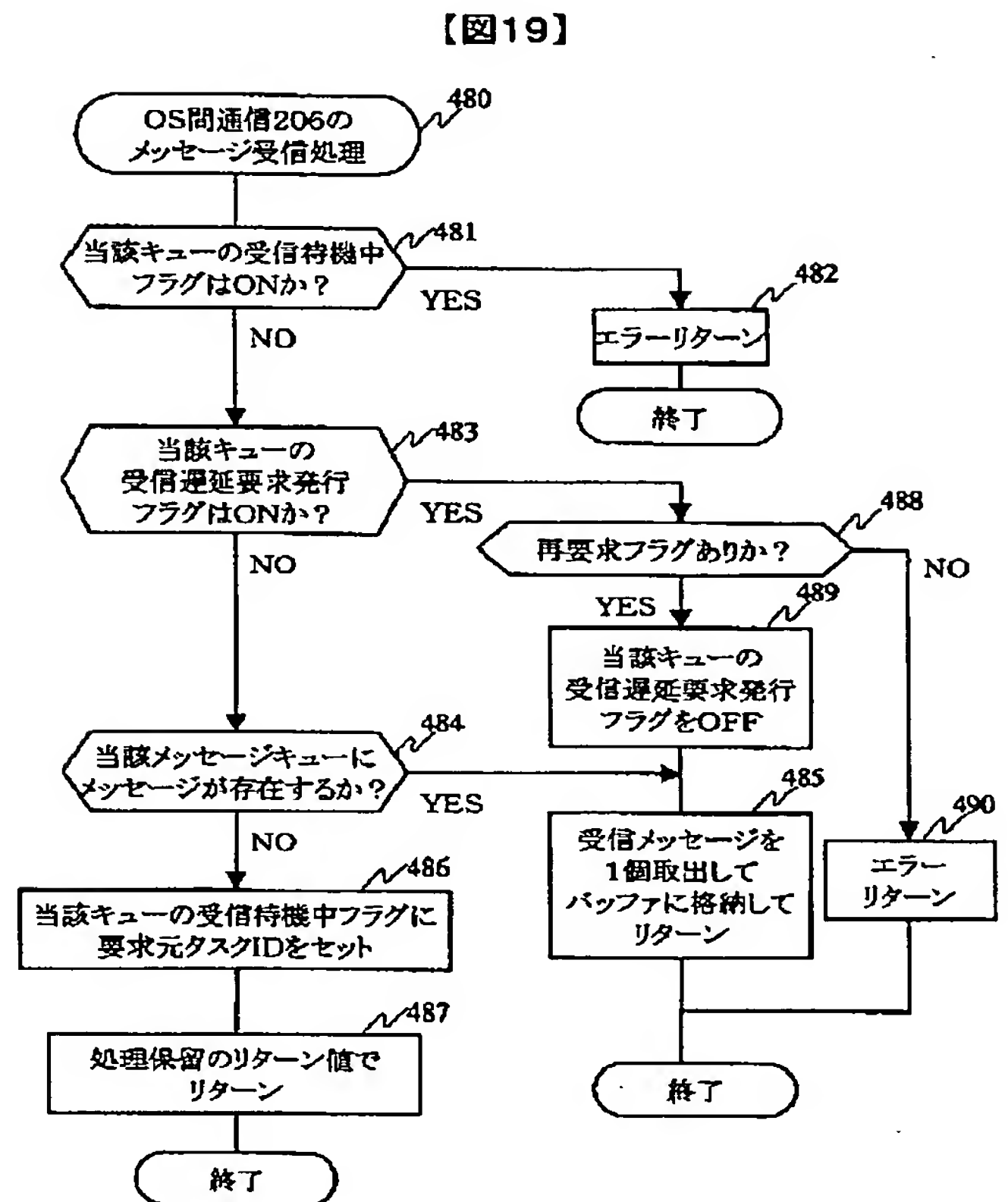
【図17】



【図18】

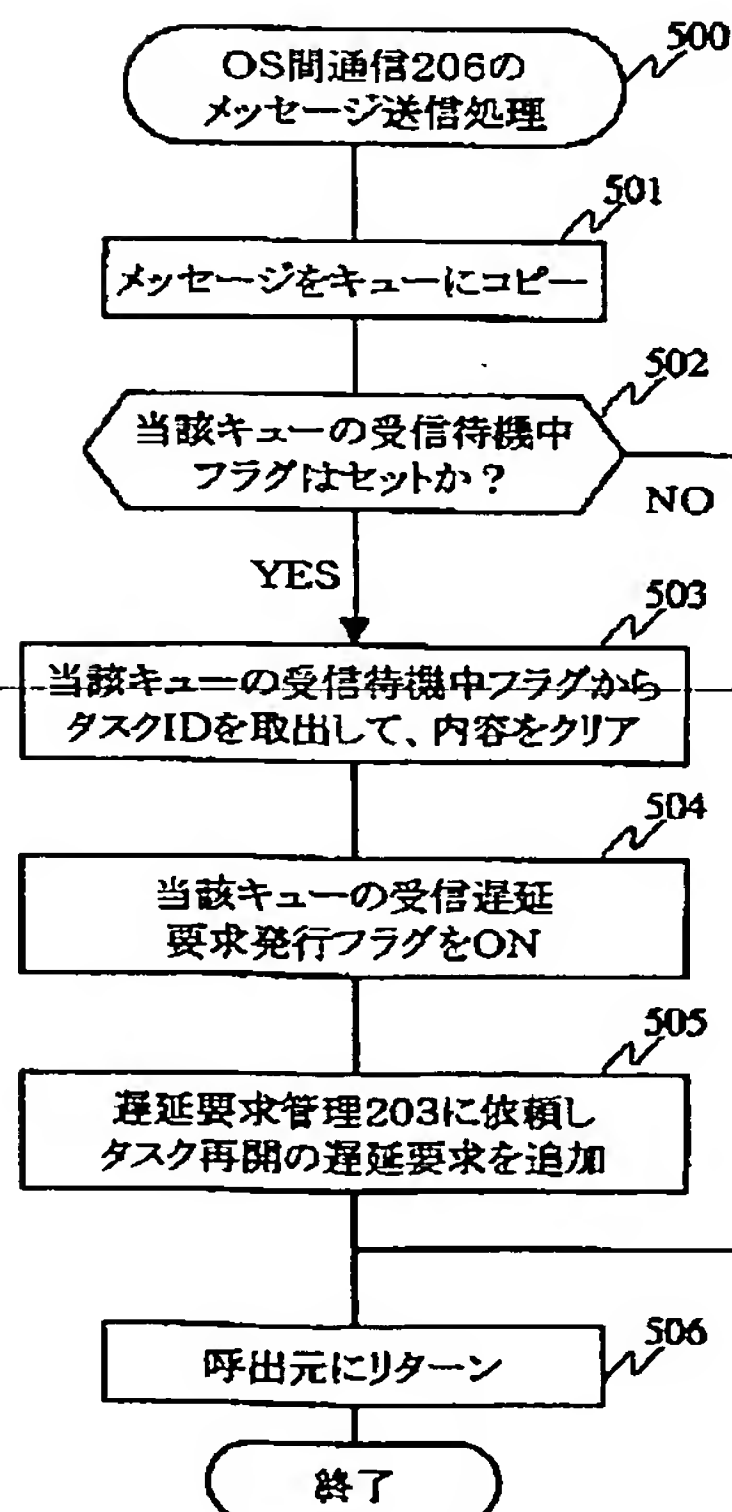


【図19】



【図20】

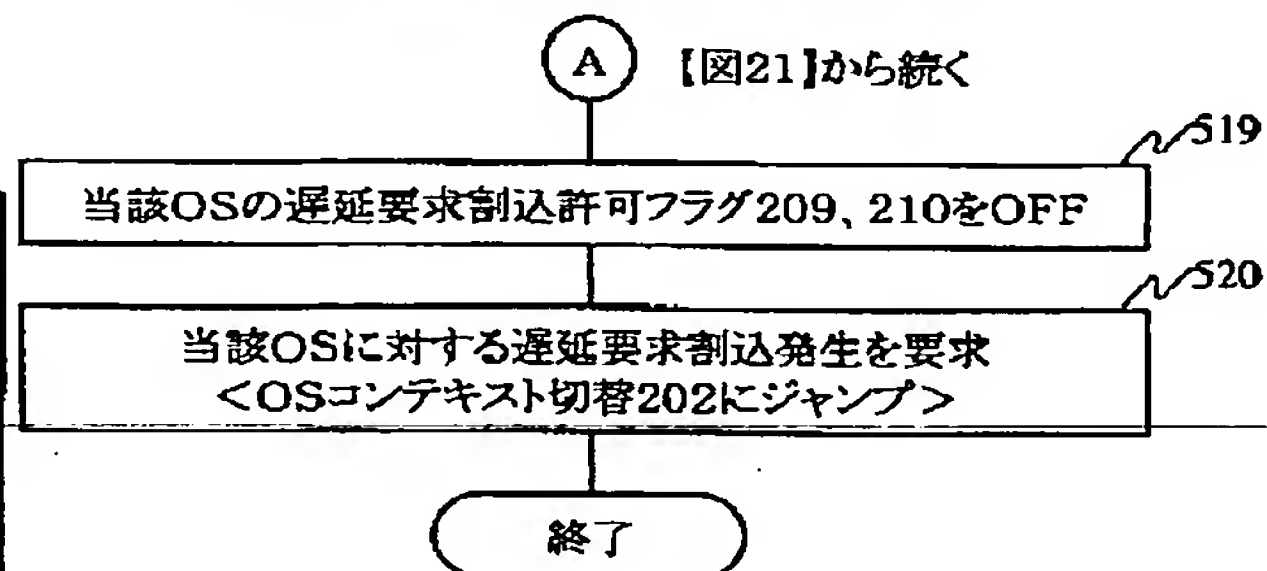
【図20】



【図22】

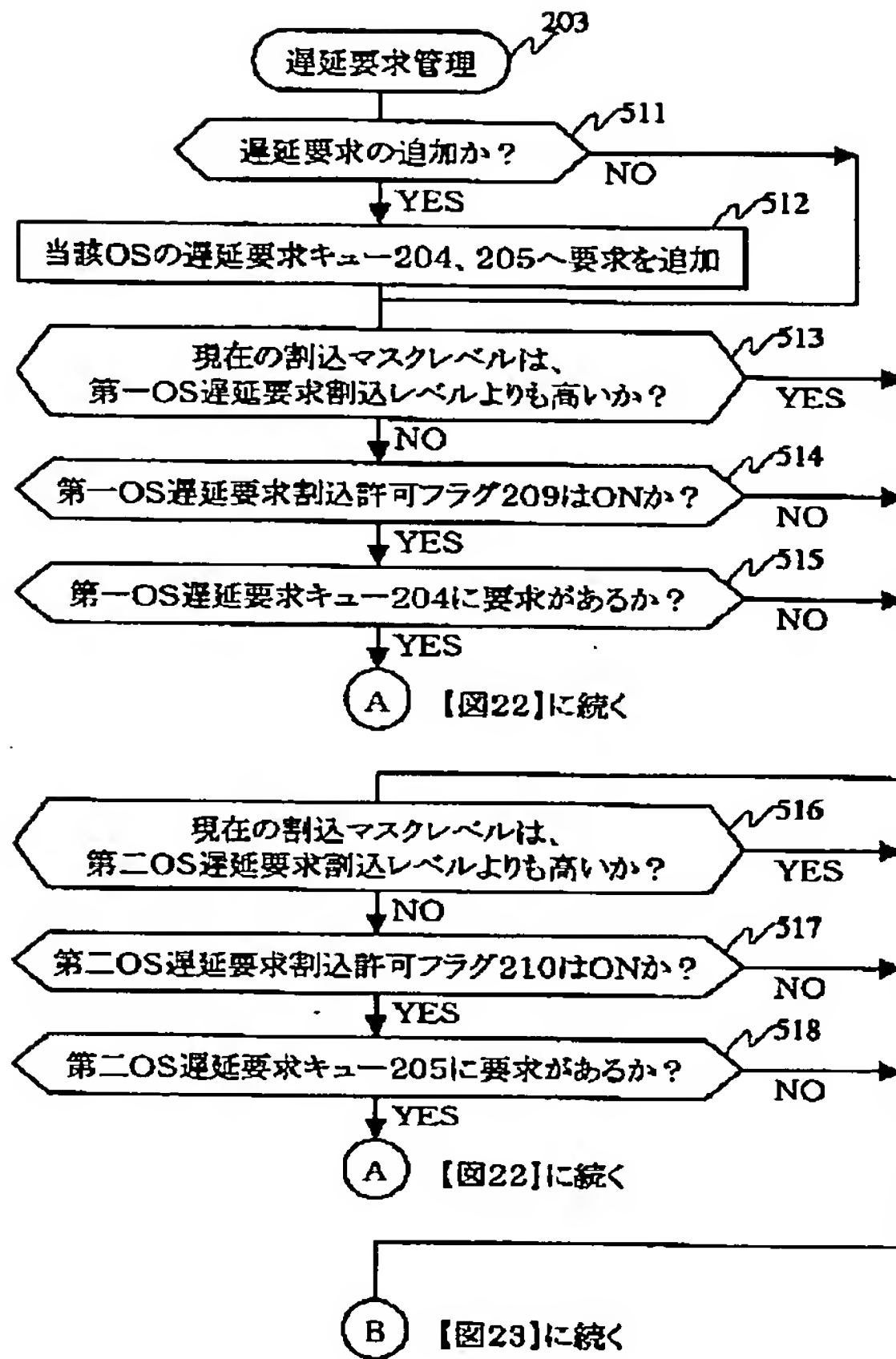
【図22】

(遅延要求割込を発行する場合の処理)



【図21】

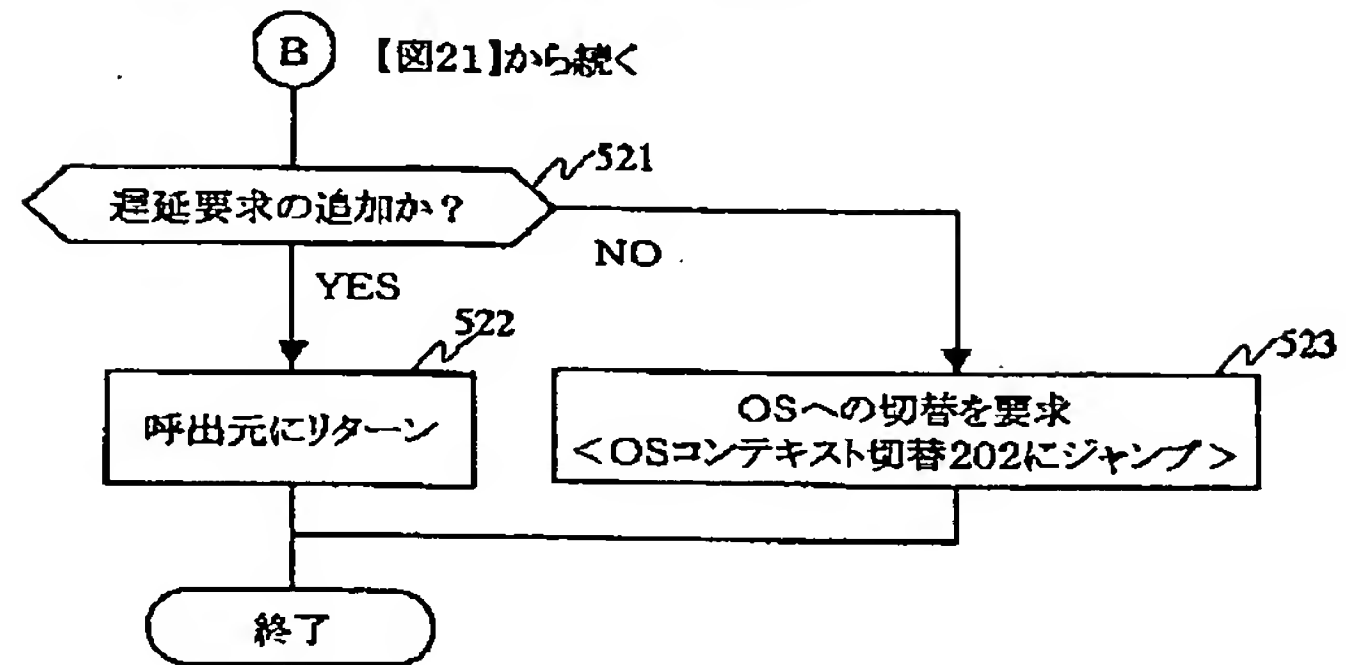
【図21】



【図23】

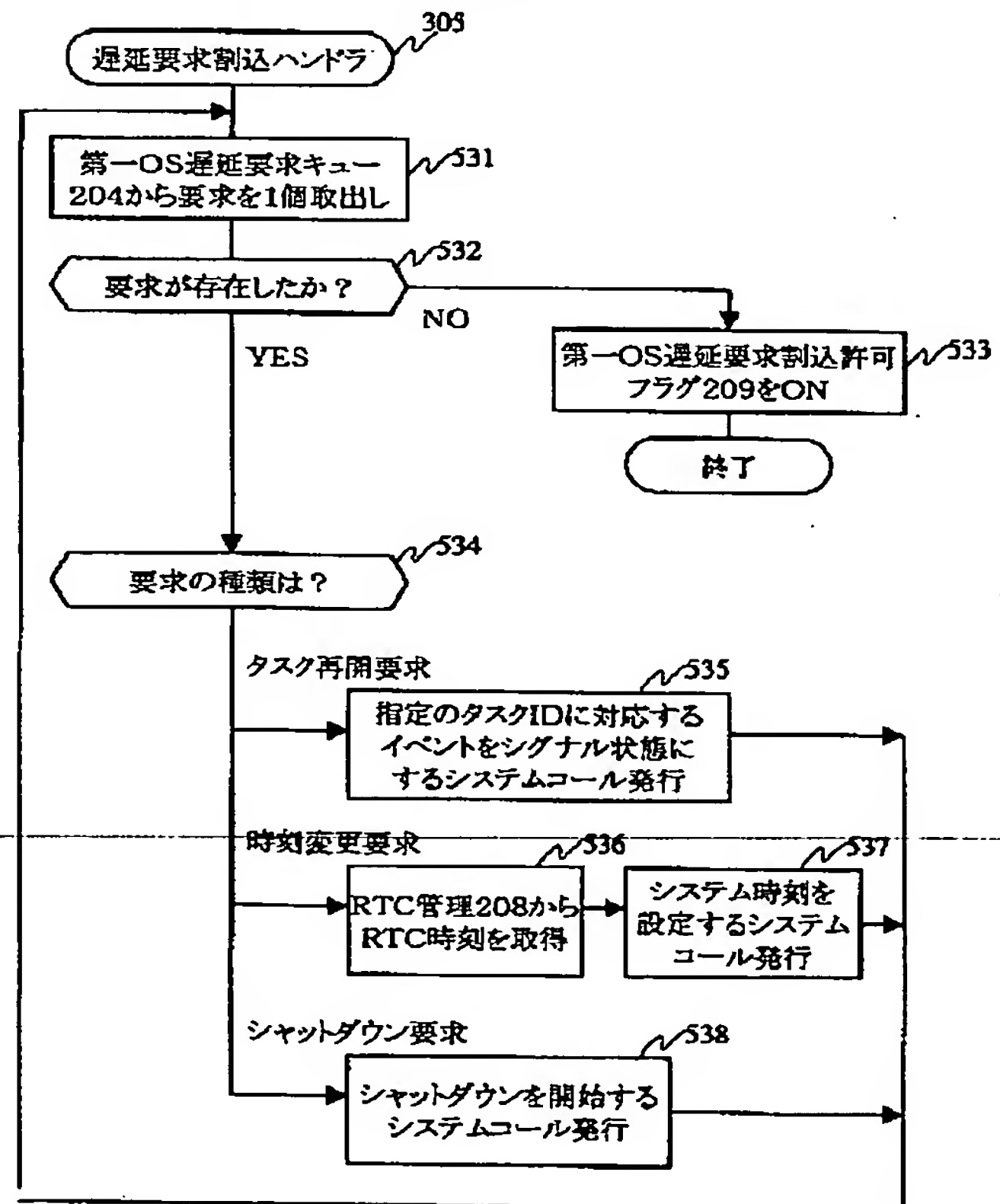
【図23】

(遅延要求割込を発行しない場合の処理)



【図24】

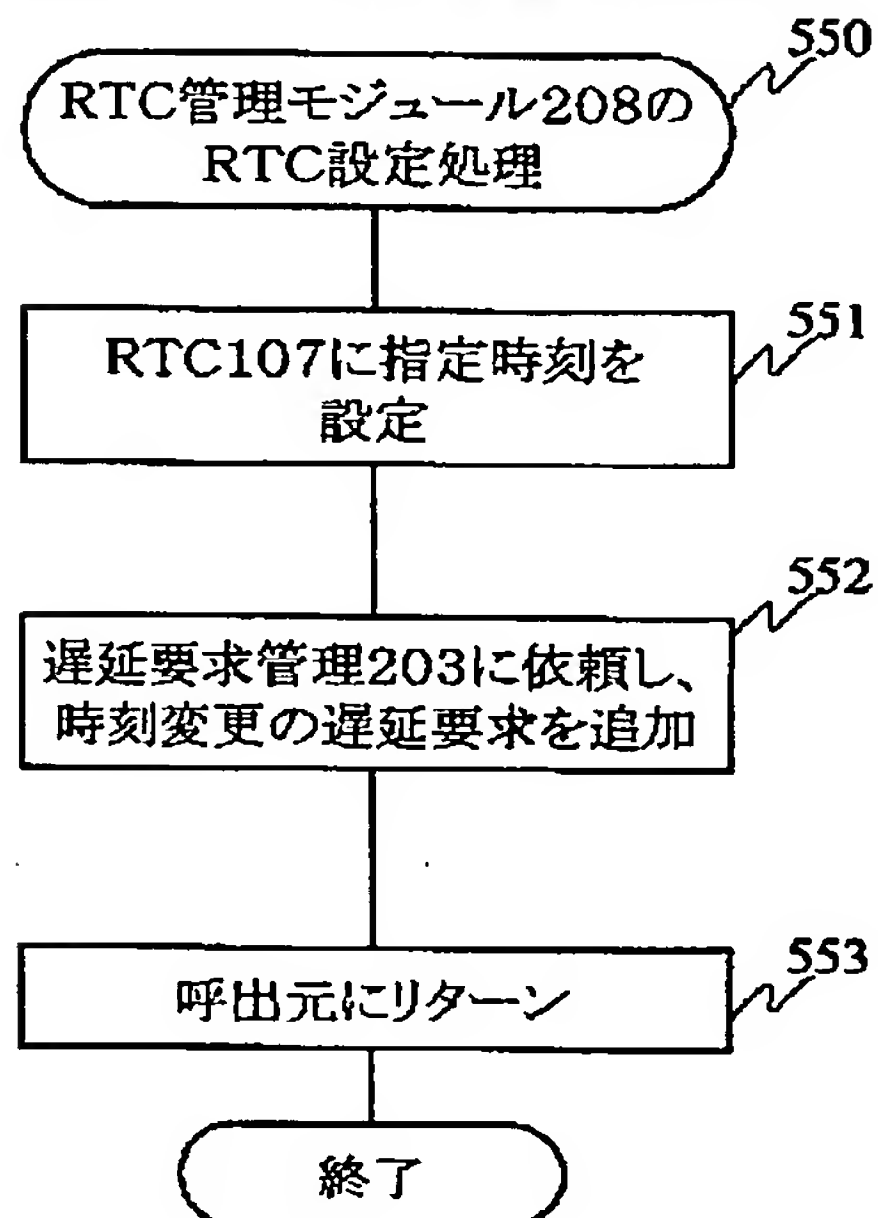
【図24】



【図25】

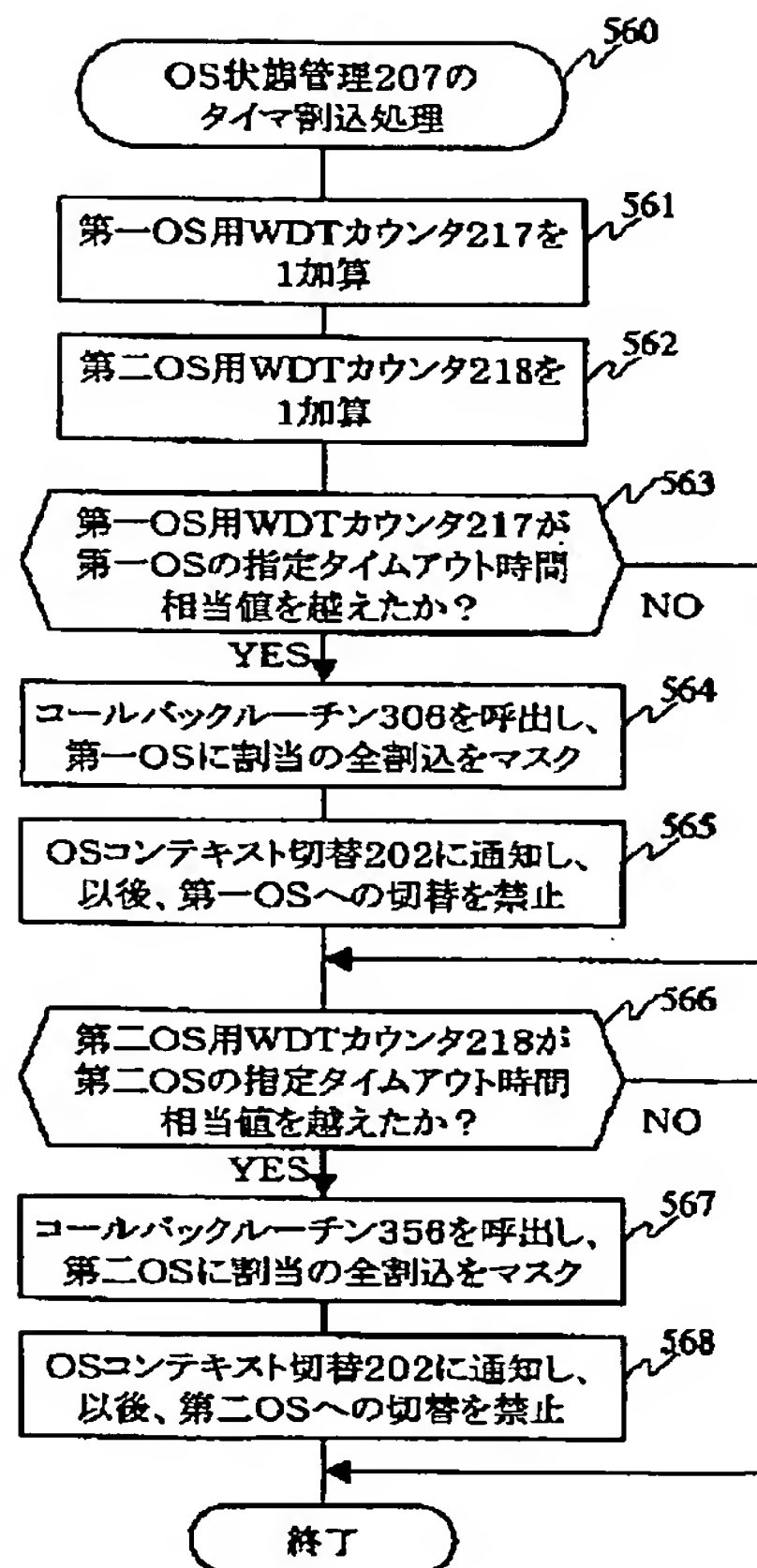
【図25】

(各OSのシステム時刻変更時に呼出)



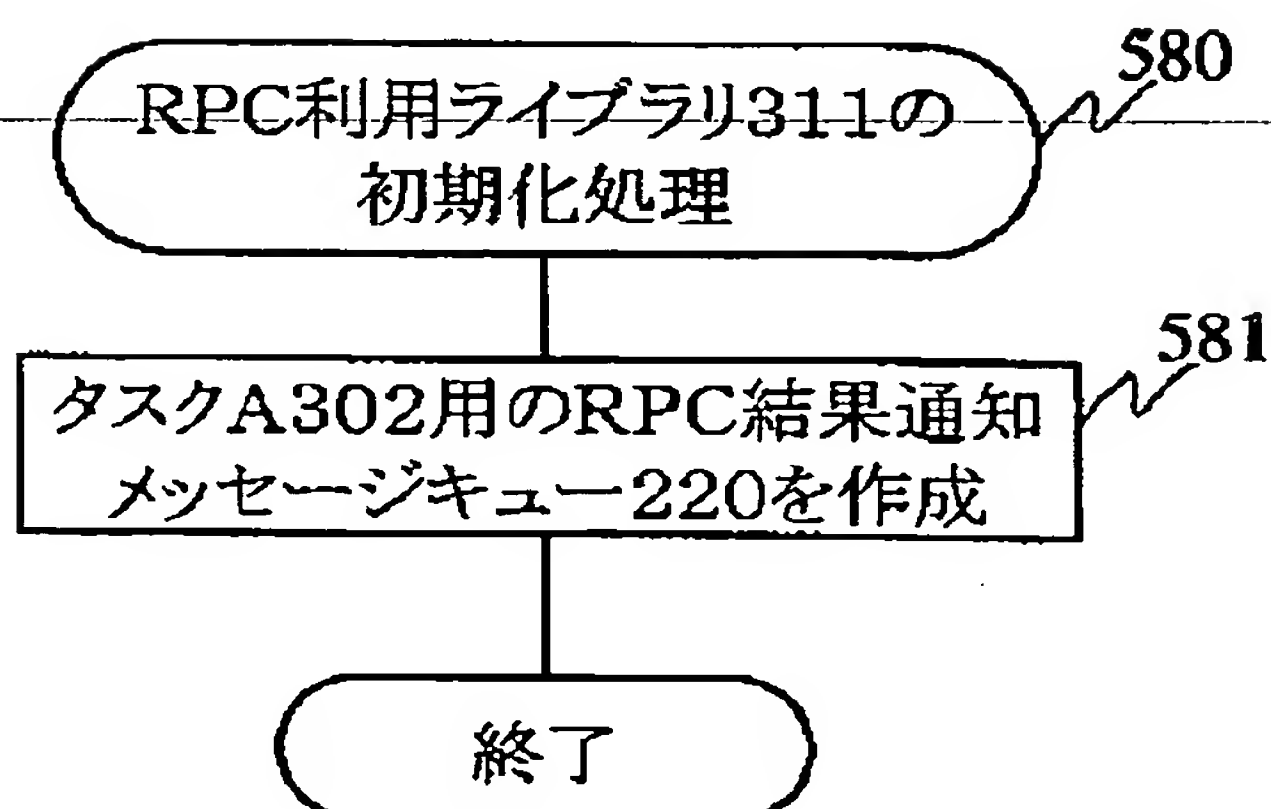
【図26】

【図26】



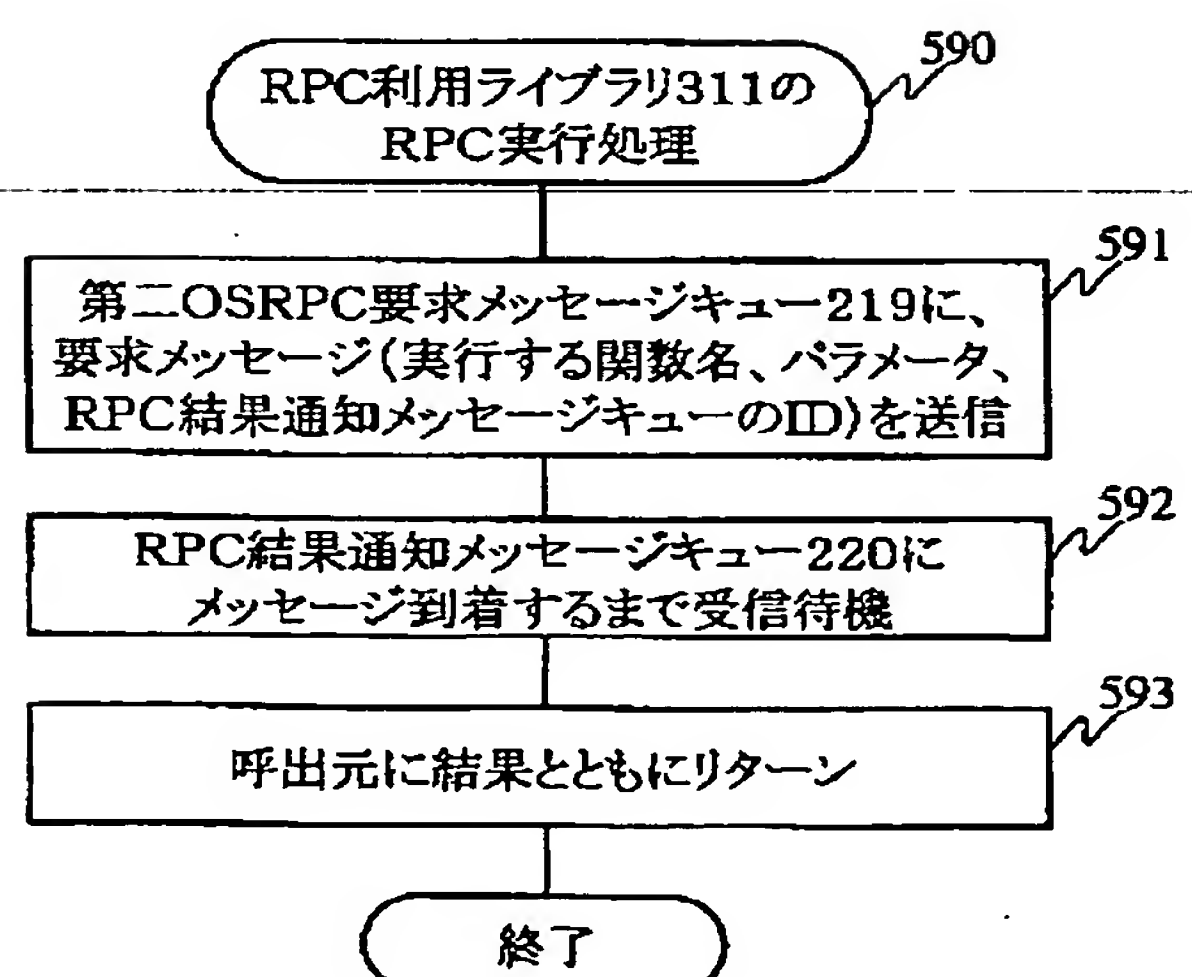
【図27】

【図27】



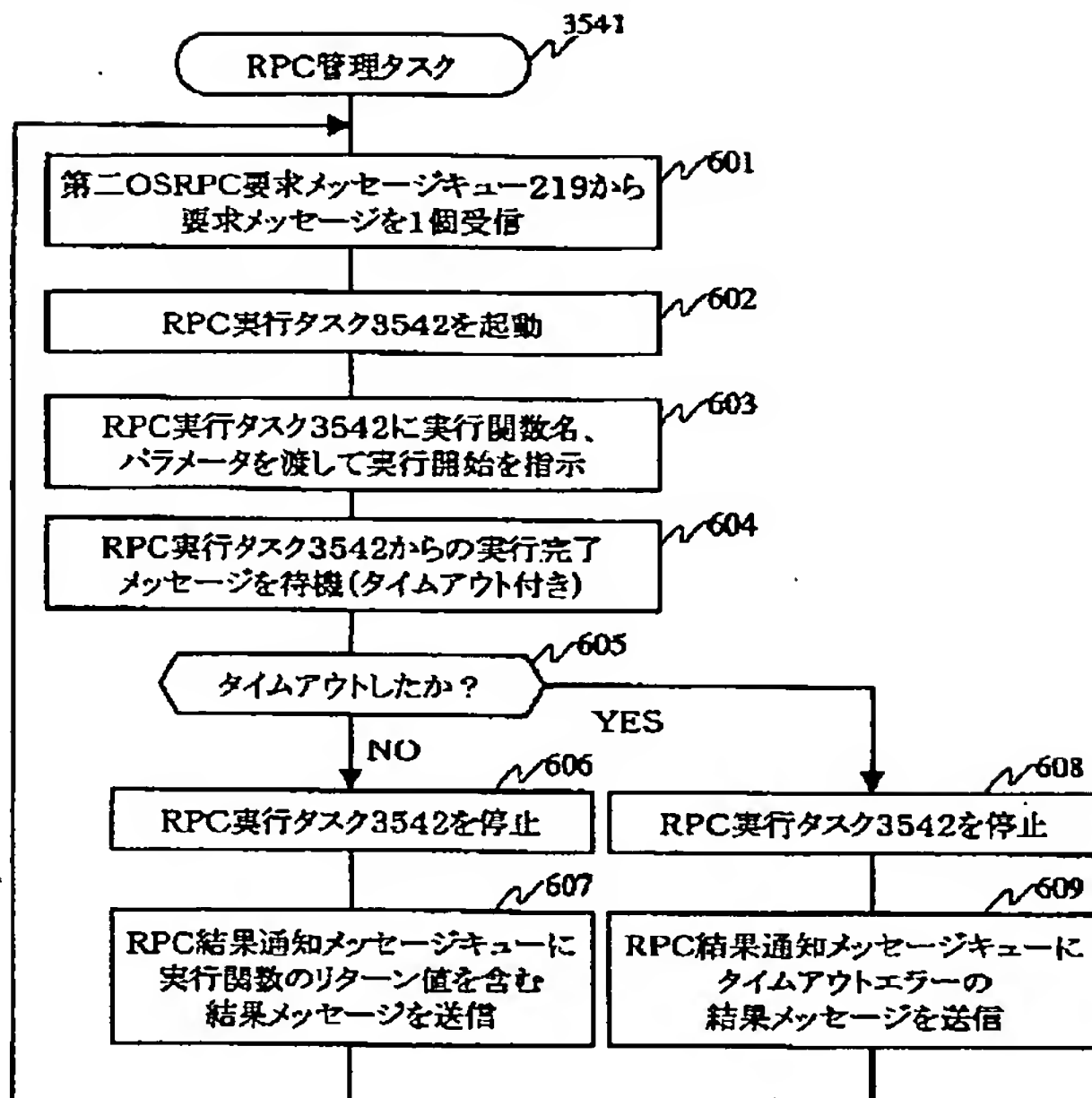
【図28】

【図28】



【図29】

【図29】



【図30】

【図30】

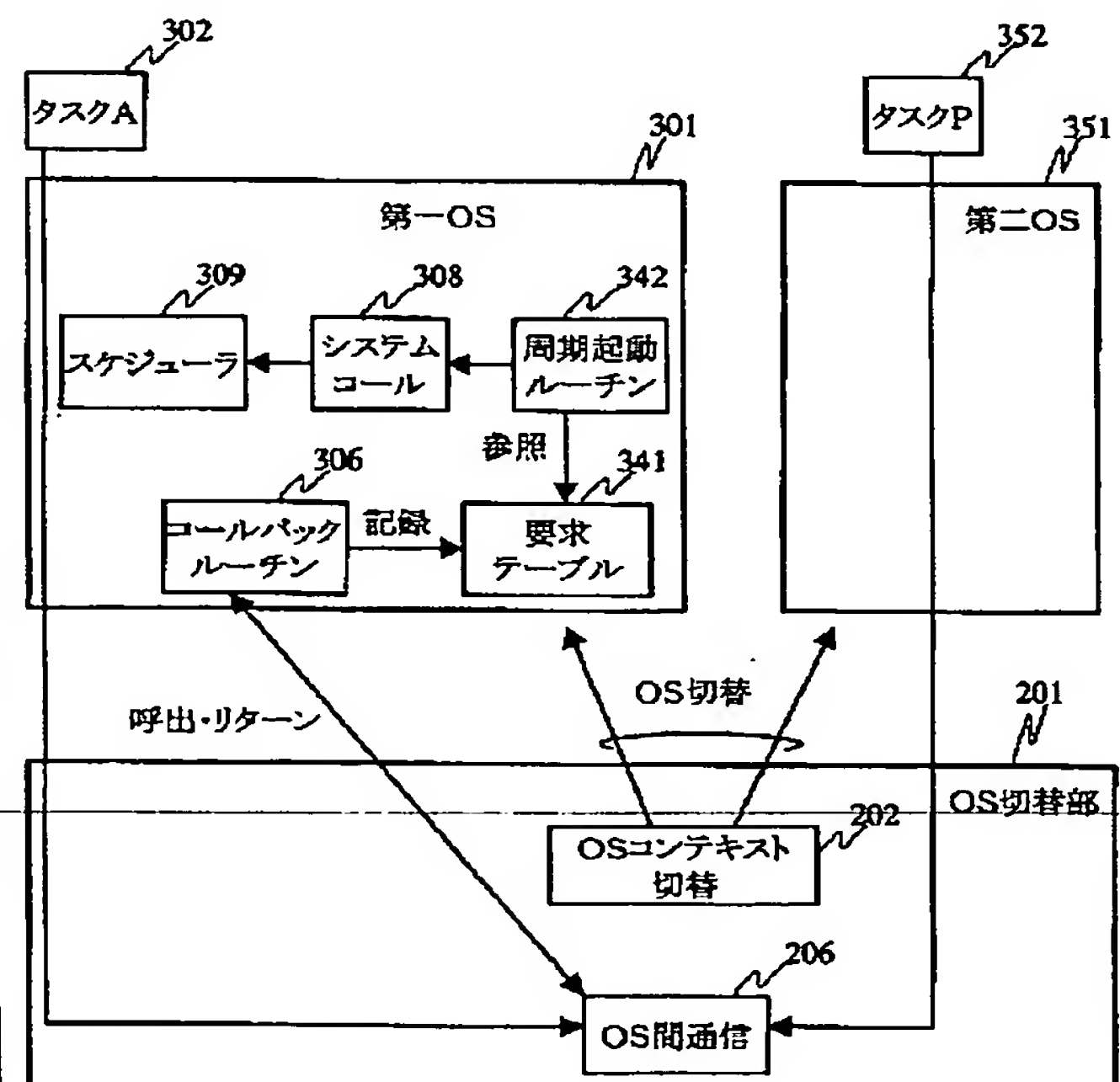
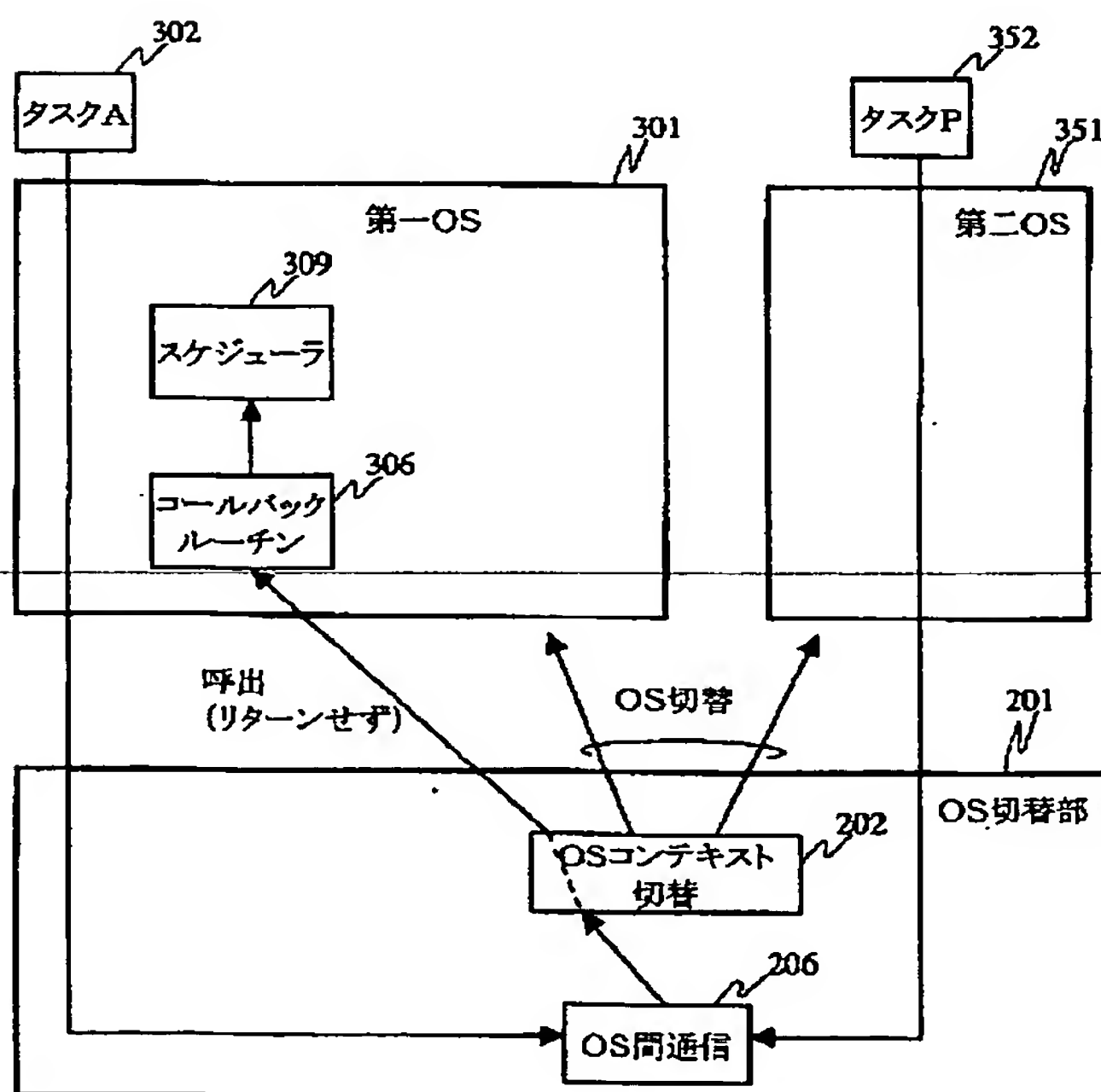
処理方法	選択基準
遅延要求	<ul style="list-style-type: none">● 処理の結果、スケジューリングが発生することがある処理。● 処理時間が長く、処理中に割込の発生やOSの切替を許可したい処理。
コールバックルーチン	<ul style="list-style-type: none">● 処理要求が発生したときに直ちに実行することが必要な処理。● 当該OSの割込処理が正常に行われていない状態で実行される可能性のある処理。● 処理の結果、スケジューリングが発生せず、処理に時間が短い処理。

【図32】

【図32】

【図31】

【図31】



フロントページの続き

(72)発明者 加藤 直
茨城県日立市大みか町五丁目2番1号 株
式会社日立製作所大みか事業所内

(72)発明者 上脇 正
茨城県日立市大みか町五丁目2番1号 株
式会社日立製作所大みか事業所内
Fターム(参考) 5B098 BA06 BB05 EE06 GA02 GA04
HH01 HH04